May 2020

Applying Statistical Mechanics to Improve Computational Sampling Algorithms and Interatomic Potentials

Mariia Karabin

Clemson University, mariya.karabin8@gmail.com

Applying Statistical Mechanics to Improve Computational Sampling Algorithms and Interatomic Potentials

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Chemistry

---

by
Mariia Karabin
May 2020

Accepted by:
Dr. Steven Stuart, Committee Chair
Dr. Brian Dominy
Dr. Leah Casabianca
Dr. Robert Latour

Abstract

In this dissertation the application of statistical mechanics is presented to improve classical simulated annealing and machine learning-based interatomic potentials.

Classical simulated annealing is known to be among the most robust global optimization methods. Therefore, many variations of this method have been developed over the last few decades. This dissertation introduces simulated annealing with adaptive cooling and shows its efficiency with respect to the classical simulated annealing. Adaptive cooling simulated annealing makes use of the on-the-fly evaluation of the statistical mechanical properties to adaptively adjust the cooling rate. In this case, the cooling rate is adaptively adjusted based on the instantaneous evaluations of the heat capacities, with the possible future extension to the density of states. Results are presented for Lennard-Jones clusters optimized by adaptive cooling simulated annealing and the classical simulated annealing. The adaptive cooling approach proved to be more efficient than the classical simulated annealing.

Statistical mechanics was also used to improve the quality and transferability of machine learning-based interatomic potentials. Machine learning (ML)-based interatomic potentials are currently garnering a lot of attention as they strive to achieve the accuracy of electronic structure methods at the computational cost of empirical potentials. Given their generic functional forms, the transferability of these potentials is highly dependent on the quality of the training set, the generation of which is a highly labor-intensive activity. Good training sets should at once contain a very diverse set of configurations while avoiding redundancies that incur cost without providing benefits. We formalize these requirements in a local entropy maximization framework and propose an automated sampling scheme to sample from this objective function. We show that this approach generates much more diverse training sets than unbiased sampling and is competitive with hand-crafted training sets[1].

Acknowledgments

I want to thank Dr. Steven Stuart for his teaching and explanations, endless support, and for being a great research advisor! Thank you so much for all your teaching and for letting me explore all these other opportunities and projects during my PhD program.

I cannot thank enough Dr. Danny Perez from Los Alamos National Laboratory, with whom I worked on a part of my dissertation research, for his guidance and help with the project, for all his support, and for being an incredible mentor!

I also want to thank Dr. Li-Ta Lo (Ollie) from Los Alamos National Laboratory, my first internship mentor, for being a life-time mentor!

Thank you to Dr. Stuart's research group and Dr. Kholodenko for their help and discussions. I also want to thank Dr. Dominy, Dr. Casabianca, and Dr. Latour for agreeing to serve on my committee.

There are many more people who turned this journey into a unique and exciting experience, and who will stay in my heart forever. I'm endlessly thankful to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computational chemistry is rapidly developing with the constant improvement of hardware and theoretical background. One of the main strengths of computational chemistry is the ability to perform calculations and provide insights on problems otherwise impossible to approach experimentally. With the use of computational tools it is possible to rationalize the behavior of the molecules at time scales from femtoseconds to seconds, covering 15 orders of magnitude, and on length scales from Ångstroms to millimeters[4]. This also allows study of the system at nonphysical conditions, which might still bring a useful perspective on the problem. This in turn requires a strong interconnection of computational chemistry with the other sciences.

More specifically, computational chemistry tools can provide insights on the time evolution of the systems (i.e. molecular dynamics), predict and refine structures, and help answer questions about the dependence of the system's properties on its structure. The list of general use cases is enormous and keeps increasing. Interestingly many tools, provided with minor modifications, can be used for solving many different problems and even in many areas. For instance, simulated annealing can be used for energy minimization problems in computational chemistry, as well as the computation of the economic equilibrium[5], in which the supply and demand are balanced.

Most of the common computational tools are powerful for a variety of systems and conditions, but also have their limitations. Among the most common ones are limited simulation time and system size. The typical time step in molecular dynamics is often limited to the order of the femtoseconds, which should be defined by the fastest motion in the system. This in turn leads to the limitations in the simulation time. Since many of the structural changes in the system happen on the order of the milliseconds or longer, this is a

serious limitation. A typical system size in molecular dynamics is on the order of $10^4 - 10^8$ atoms, due to the expensive atomistic interactions, for which the computational cost could be roughly proportional to the number of atoms[6].

One way around these limitations is to use purely computational approaches such as parallel computing, algorithmic and hardware improvements. Supercomputers greatly improve simulation capabilities, and especially supercomputers specifically designed to run molecular dynamics simulations such as Anton, which increases the simulation time up to around milliseconds[6].

Another way to approach these limitations is by using statistical mechanics. For instance, many processes of interest that are happening beyond roughly one microsecond are inaccessible to molecular dynamics, but can be simulated with the accelerated molecular dynamics methods (AMD), reaching the time scale sometimes beyond milliseconds[4]. AMD methods are using statistical mechanics and mainly transition state theory to improve molecular dynamics and reach longer time scales. In hyperdynamics the energy minima are filled with a bias potential to make the escapes occur more frequently and simulate long trajectories[7]. Temperature accelerated dynamics method makes use of the elevated temperature of the system to increase the escape rate[8]. In parallel replica dynamics method the time evolution of the system is parallelized by simulating many replicas in parallel[9]. The AMD methods mentioned above are highly useful when there are infrequent transitions between states and for generating long-time trajectories.

Biased sampling methods make use of the modified potential energy function to improve the sampling of the rare states and achieve broader sampling. These methods can be used with both Monte Carlo and molecular dynamics methods[10]. Biased potentials can take the form of harmonic potentials, or any other functional form. For instance, umbrella sampling method is highly useful when certain states are separated by extremely high energy barriers. Umbrella sampling most commonly uses a harmonic potential as a weighting function to modify the potential energy function. The generated distribution is non-Boltzmann, but to relate back to the usual ensembles and calculate Boltzmann averages one can use re-weighting techniques[11, 12].

Many other tools, and not just the ones mentioned above, can be improved using statistical mechanics. In this dissertation, we present two cases where statistical mechanics can be used to improve a computational sampling method (and more specifically classical simulated annealing) and the automated generation of training sets for machine learning-based interatomic potentials.

In Chapter 3 an approach for improving the classical simulated annealing (SA) method using statistical mechanics is introduced. In this approach an on-the-fly evaluation of the heat capacity is used to decide on the cooling rate, which should be slow during the temperature region when the heat capacity is above a

certain heat capacity cutoff value, and fast cooling should be chosen for heat capacity values below the cut-off. This approach results in an adaptive cooling simulated annealing (ACSA) method, that is more efficient than the classical simulated annealing, particularly for systems with many degrees of freedom. Portions of this chapter have been published on arXiv[13] and are in the process of being submitted for peer-reviewed publication.

In Chapter 4 another approach is introduced in which statistical mechanics is used to improve the generation of the training sets for machine learning-based interatomic potentials. In this method an entropy-maximization approach is used to improve the (sometimes highly labor-intensive) generation of the training sets. The training sets generated using this approach contain a more diverse set of configurations and improved transferability, when compared to the manually generated training set used for developing tungsten potentials[14]. This chapter has been published on arXiv[1] and is in the process of being submitted for peer-reviewed publication.

Chapter 2

Methods

2.1 Molecular Dynamics

Molecular dynamics (MD) methods are used to explore the evolution of a system with time by solving Newton's equations of motion for this system, and provide kinetic and thermodynamic information on a time scale otherwise inaccessible experimentally. These methods emerged during the 1950s[15, 16], starting with the use of the hard-sphere model and modeling single molecules in vacuum, and have evolved tremendously since then towards modeling much more realistic conditions and complex systems like proteins. Molecular dynamics provides information such as atomic velocities, positions and energies, and then statistical mechanics is used to connect microscopic and macroscopic observables. Therefore, MD is used in many areas including structure prediction, molecular design and many more.

Molecular dynamics makes use of the Born-Oppenheimer approximation and therefore each atom is represented as a point mass. The instantaneous forces are calculated iteratively during the simulation and particles move in accordance with the equations of motion. To obtain the trajectories, Newton's equation of motion needs to be solved:

$$\frac{d^2x_i}{dt^2} = \frac{F_{x_i}}{m_i} \tag{2.1}$$

where m_i is the mass of atom i with the coordinate x_i and F_{x_i} is the force exerted on that atom in that coordinate. Knowing the position $x(t)$ at a time t, the positions after a time step δt can be expressed using Taylor

series:

$$x(t+\delta t) = x(t) + \frac{dx(t)}{dt}\delta t + \frac{d^2x(t)}{dt^2}\frac{\delta t^2}{2} + \ldots \tag{2.2}$$

The higher terms for the Taylor expansion can be truncated at the acceleration (second derivative), and the assumption is that the higher order terms can be neglected. Many finite difference methods have been developed for the integration of the equations of motion. Ideally, we want to have an integrator that allows the longest time step possible with as few force calculations as possible. The force calculations are usually the most computationally expensive part of a molecular dynamics simulation. One of the widely used integrators is the velocity Verlet method[17]:

$$x(t+\delta t) = x(t) + \delta t v(t) + \frac{1}{2}\delta t^2 a(t) \tag{2.3}$$

$$v(t+\delta t) = v(t) + \frac{1}{2}\delta t[a(t) + a(t+\delta t)] \tag{2.4}$$

To calculate the velocities at $t+\delta t$ this method requires the accelerations at $t+\delta t$ as well, and therefore has to be implemented in three stages[11]. First, we need to calculate new positions $x(t+\delta t)$ (eq. 2.3) and intermediate velocities:

$$v(t+\frac{1}{2}\delta t) = v(t) + \frac{1}{2}\delta t a(t) \tag{2.5}$$

Now we can re-compute the forces, given the updated positions, from which we get the accelerations $a(t+\delta t)$, and calculate the final velocities:

$$v(t+\delta t) = v(t+\frac{1}{2}\delta t) + \frac{1}{2}\delta t a(t+\delta t) \tag{2.6}$$

The velocity Verlet method is time-reversable, requires only one force calculation per time step, is fairly easy to implement, and gives the accuracy up to the order of $O(\delta t^4)$.

Another important aspect of the MD method to consider is the choice of the time step. Too large time step may cause instabilities, leading to conservation errors, and the trajectory may highly diverge. If the time step is too small, it will limit the exploration of the phase space and require much more computational time.

Ideally, the optimal time step provides a balance between the correct trajectories and the rapid exploration of the phase space, and should be roughly an order of the magnitude smaller than the fastest motion[11].

2.2 Classical Simulated Annealing

Simulated annealing is a highly robust optimization method and is used in Chapter3 to optimize Lennard-Jones clusters and as a benchmark against which the new ACSA method is compared. This method was inspired by the annealing processes in metals. The process starts at a high temperature, to ensure that the metal is in a liquid state, and then the temperature is gradually reduced at a certain cooling rate, in accordance with a specific cooling schedule. The crystal structure (ground state) will be achieved if a sufficiently slow cooling rate is used. If the cooling rate is too fast, the metal atoms will end up in a glassy or crystalline-like (meta-stable) structure[18].

This idea is implemented in the classical simulated annealing method, in which the optimization begins at a high temperature T_i, and then the temperature is gradually reduced at a certain cooling rate k, in accordance with an exponential cooling schedule:

$$T(t) \quad = \quad T_i e^{-kt} \tag{2.7}$$

If the cooling rate provided at the beginning of the optimization is too fast, then the system will likely be trapped in one of the local minima (i.e. a crystalline-like structure in the case of a metal). Too slow cooling will require much more optimization time, but will also provide better optimization results. The challenge is to provide the fastest cooling rate that still results in a highly accurate optimization.

Simulated annealing can be used in combination with molecular dynamics (see section 2.1) or the Metropolis Monte Carlo algorithm[19] for sampling purposes. In case with the Metropolis Monte Carlo algorithm, a possible state is randomly generated and then the acceptance criterion is applied:

$$p(\Delta E) \quad = \quad \begin{cases} e^{\left(-\frac{\Delta E}{k_B T}\right)}, & \Delta E > 0 \\ 1, & \Delta E \leq 0 \end{cases} \tag{2.8}$$

where k_B is the Boltzmann constant, and $\Delta E = E_j - E_i$ is the difference between the energy of the new state E_j and the current state E_i. If $\Delta E \leq 0$, i. e. the energy decreases, then the new state is accepted with the probability of 1. If $\Delta E > 0$, i. e. the energy increases, then the new candidate state can be accepted with

the probability of $e^{\left(-\frac{\Delta E}{k_B T}\right)}$. The temperature is gradually decreasing as the new state is accepted and the probability for transitions with $\Delta E > 0$ to be accepted decreases. This way the downhill moves are becoming more favored. Therefore, the optimization begins at a high temperature, at which most of the uphill moves (increases in energy) would be accepted, to avoid getting trapped in one of the local minima. In case with the molecular dynamics, the general idea is the same, except instead of the acceptance probability, the uphill and downhill moves depend on the dynamics. Again, the probability of crossing energy barriers is high at first, and decreases as the temperature is reduced.

2.3 Simplex optimization

A simplex algorithm is an efficient optimization method that is used in Chapter3 to optimize parameters for classical SA and ACSA. A simplex is a geometrical figure, which depends on the dimensionality D of the objective function and has D+1 interconnected vertices[20]. Therefore, if the dimensionality is two, the simplex will have a triangular shape. The algorithm was originally developed by Nelder and Mead for function minimization[21].

In the downhill simplex algorithm, the simplex moves around on the objective function surface in search for the minimum with three basic movement options: reflection, reflection with expansion, and contraction in one or all directions. For example, in a search for a lower function value, a common move is the reflection of the highest function value through the opposite face. If this function value from the reflection move turns to be the lowest one, then the next possible simplex move is the reflection with expansion of the current highest function value[11], with subsequent contraction when it's close to the local minimum. The highest function value is the one that is always being replaced. This way simplex is adaptively adjusting to the landscape.

Consider an n-dimensional space, in which $P_0, P_1, \ldots P_n$ are the current simplex points. Let y_i correspond to the function value at the point P_i. Therefore, the highest y_h and lowest y_l function values are[21]:

$$y_h = \max_i(y_i) \tag{2.9}$$

$$y_l = \min_i(y_i) \tag{2.10}$$

The reflection move is defined as:

$$P^* = (1+\alpha)\overline{P} - \alpha P_h \tag{2.11}$$

where $\overline{P}$ is the centroid of the points for which $i \neq h$, α is the reflection coefficient, and P_h is the point with the highest value. If the function value y^* that corresponds to P^* is lower than y_h then P_h is replaced with P^*. An expansion move is performed if y^* is lower than y_l:

$$P^{**} = \gamma P^* + (1-\gamma)\overline{P} \tag{2.12}$$

where γ is the expansion coefficient. Similarly to the reflection the point P^{**} is accepted if the function value y^{**} is lower than y_l. In the contraction move:

$$P_{con} = \beta P_h + (1-\beta)\overline{P} \tag{2.13}$$

where β is the contraction coefficient. The contraction fails if the function value $y_{con} > \min(y_h, y^*)$. Lastly, in the collapse move all point are replaced:

$$P_{coll} = 0.5(P_i + P_l) \tag{2.14}$$

where P_i is the simplex point for which $i \neq l$, and P_l is the point with the lowest value. The individual sequence of moves is chosen with an algorithm described by Nelder and Mead[21]. In this case, the simplex optimization continues until the calculated tolerance value (eq. 2.15) is below a pre-determined cutoff value.

$$tol = \frac{2|y_h - y_l|}{|y_h + y_l|} \tag{2.15}$$

2.4 Spectral Neighbor Analysis Potential

The spectral neighbor analysis potential (SNAP)[22] is a machine-learning based potential that is used in Chapter 4 to generate descriptors to construct our abstract potential energy function. This potential computes the energy of an atom based on the density of neighboring atoms in spherical shells surrounding the atom and has a very general and non-fixed functional form, which allows the modeling of a variety of

systems and can be used for liquids and solids. Machine-learning techniques, and in this case weighted least-squares linear regression, are used with the quantum mechanical results as a reference to determine SNAP coefficients. The coefficients can be improved by using more data from quantum electronic structure calculations.

The atomic environment in SNAP is described using bispectrum components, which are the descriptors for the local structures, and have been previously developed for the Gaussian approximation potential (GAP) [23]. These descriptors are based on the neighbor density around a central atom i in a spherical shell, which can be expanded using the hyperspherical harmonics as the basis functions. The expansion coefficients of the neighbor density are combined in a way that makes them rotationally invariant. This results in the bispectrum components which have been used as the descriptors in SNAP.

A more detailed description of the local structures can be obtained by including higher order bispectrum components, while the coarse features can be described with just the low-order descriptors. The cost of calculation of the bispectrum components increases for the higher order descriptors. Therefore, the number of descriptors usually used is 6 - 100. These bispectrum components are then used to express the total energy, assuming a linear relationship between the energy and the bispectrum components, as:

$$E_{SNAP}(r^N) = N\beta_0 + \beta \sum_{i=1}^{N} q^i \tag{2.16}$$

where β_0 is the energy contribution for an isolated atom, β is a vector of SNAP coefficients, and q^i is a vector of bispectrum components. The energy of atom i is expressed with a set of bispectrum components:

$$E^i_{SNAP}(q^i) = \beta_0^{\alpha_i} + \sum_{k=1}^{K} \beta_k^{\alpha_i} q_k^i \tag{2.17}$$

where $\beta_k^{\alpha_i}$ is the *k-th* linear coefficient for atom i of type α, and q_k^i is the *k-th* bispectrum component for atom i from a set of K descriptors. Therefore, the results can be extended for the different atom types. This linear relationship between atom energy and bispectrum components makes it easier to find a best set of SNAP coefficients, rather than for the potentials without a linear relationship between the properties of interest and their descriptors. Therefore, one could construct a set of linear equations, which would be solved for the SNAP coefficients β using weighted least-squares linear regression. When fitting the SNAP potential to the large reference data set obtained from DFT calculations, these optimal SNAP coefficients minimize the difference between the energies obtained from the SNAP potential and the energies from the reference data

set.

The force is expressed with the derivatives of the bispectrum components:

$$F^j_{SNAP} = -\nabla_j E_{SNAP} = -\beta \sum_{i=1}^{N} \frac{\partial q^i}{\partial r_j} \tag{2.18}$$

where F^j_{SNAP} is the force on atom j. These forces can then be used to perform molecular dynamics simulations with the SNAP model.

Chapter 3

Simulated Annealing with Adaptive Cooling Rates

The text of this chapter, except for section 3.3, will be submitted to J. Chem. Phys.: M. Karabin and S. J. Stuart, "Simulated Annealing with Adaptive Cooling Rates"[13].

3.1 Introduction

Simulated annealing (SA) is one of the most robust optimization methods, and has been widely used for global optimization and energy minimization problems for decades[24]. The method makes use of Boltzmann sampling of an energy landscape at a temperature which is gradually reduced[11, 25]. At elevated temperatures, the system has enough energy to be able to cross energy barriers and find the basins that contain the important energy minima, and at low temperatures (with suitably slow cooling) it converges to a neighborhood around the global minimum.

The simulated annealing method is widely used, due to its simplicity and ease of implementation, but suffers from being very time consuming[26]. It is guaranteed to converge to the global minimum only in the limit of infinitely slow cooling rates, so a very gradual decrease in temperature gives the most robust results, but is also computationally costly. If the temperature is decreased too rapidly, however, the system is likely to become trapped in a local minimum[27].

For this reason, considerable effort has been devoted to improving the efficiency of the SA algorithm,[18]

based on improvements to the cooling schedule, learning mechanism, and neighborhood selection. For example, Monte Carlo sampling has been performed with wide-tailed distributions for sampling states,[28] hybrid methods have been developed that extend the Boltzmann or Monte Carlo sampling used in vanilla SA to include sampling based on the genetic algorithm[29], differential evolution[30], particle swarm optimization[31], and the harmony search algorithm[32].

Independently from the method used to sample states in configuration space, the cooling schedule used to control the annealing temperature can also have an effect on the efficiency of the optimization. The goal is to design the fastest cooling schedule that still results in an acceptably high probability of convergence to the global minimum. Much effort has been devoted to studying whether the optimal cooling schedule is linear in time, or exponential[33], proportional[25], nonlinear[34], inversely linear[28], logarithmic[35], cooling schedules that incorporate variance of temperatures[36], geometrical[37], or adaptive[38].

Adaptive cooling schedules use information about the system during execution such as energy variance or temperature variance to adjust the cooling[18].

The adaptive simulated annealing method is a variation of the very fast simulated re-annealing[39]. It has an improved cooling schedule and takes advantage of the simulated quenching but does not assure global optimum.

Unfortunately, the answer to the question of which cooling schedule is optimal depends on the system. Systems with a folding funnel-style energy landscape can often tolerate rapid cooling, while systems with a rugged energy landscape and a broad distribution of energy barriers may need a more conservative cooling schedule. Slow cooling is most important at temperatures where the thermal energy kT (just barely) allows crossing of the critical barriers allowing escape from important local minima. Such detailed knowledge of the energy landscape is often unavailable when performing an optimization, of course, so the ideal cooling schedule is typically unknown.

An ideal cooling schedule would cool quickly during portions of the optimization when the system is unlikely to become kinetically trapped in a local minimum, but much more slowly at key temperatures where kinetic traps are more accessible. Unfortunately, this idealized approach requires detailed knowledge of the features of the function being optimized — features which are generally not known if the function is being optimized.

What is needed is a method that can adaptively determine the instantaneous cooling rate, based on the energetic properties of the system being simulated, but without making use of any a priori knowledge of the features of the energy landscape. The purpose of this chapter is to describe such a method, using on-

the-fly statistical mechanical property evaluation to adaptively adjust the cooling rate in real time during the optimization. Ideally, these properties would include a weighted distribution of barrier heights accessible at the current thermal energy, but this requires non-local knowledge of the potential energy surface. As a proxy for this information, we use the heat capacity of the system, which is large when the temperature is at a value that (just) allows exploration of a range of new energy basins.

The reason for choosing a slow cooling rate when the instantaneous value of the heat capacity is higher than the heat capacity cutoff, is to ensure that the system spends proportionally more time at temperatures where it has just enough energy to cross the energy barriers. This is when the slight change in temperature and slower cooling rate results in a better exploration of the phase space. By providing a slow enough cooling we reduce the probability that the system gets stuck in a local minima. When the instantaneous value of the heat capacity is lower than the heat capacity cutoff, the system is cooled down at a much faster cooling rate, which decreases the optimization time. The instantaneous value of the heat capacity is lower than the heat capacity cutoff at the high temperatures used at the very beginning of the simulation, when we can use a faster cooling rate until conditions such that a slight change in temperature makes a great impact on the exploration of the phase space. This is when our instantaneous value of the heat capacity becomes higher than the heat capacity cutoff, and a slower cooling rate is used. At lower temperatures, when the system becomes effectively trapped in a local (and hopefully the global) minimum, the instantaneous value of the heat capacity becomes lower than the cutoff value, and a faster cooling rate is used again.

Consider the case of a phase change, such as the liquid-solid transition. To anneal a liquid into the global-minimum solid configuration, rather than one of the many local-minimum glassy configurations, it would suffice to cool very rapidly to just above the melting temperature, then very slowly across the phase transition, and then very rapidly thereafter. The heat capacity becomes very large near the phase transition temperature, so this can be used as a signature that slow cooling is needed, even when the phase transition temperature is unknown ahead of time. Note, also, that the cooling rate varies quite non-monotonically, alternating between slow and fast cooling rates. Even in the absence of a bona fide phase transition, the heat capacity will be larger at temperatures where more states are becoming thermally accessible, and these are the temperatures where a SA algorithm should be cooling most slowly.

3.2 Methods

In the most basic implementation of SA,[25] the temperature is lowered from some initial temperature, T_i, to a final temperature, T_f, using an exponential cooling schedule,

$$T(t) \quad = \quad T_i e^{-kt}, \tag{3.1}$$

with a constant cooling rate k. The varying temperature is used to perform Boltzmann sampling of the states of the system. Both the success and the efficiency of the optimization depend strongly on this cooling rate; when k is too large, the system will quench into a non-global minimum with an unacceptably large probability, but when k is too small, the optimization will be unacceptably slow to complete. Unfortunately, k must be chosen before the optimization begins, and usually before much is known about the distribution of local minima on the energy landscape. Consequently, k is usually treated as a purely empirical parameter; it is typically chosen to be as small as can be computationally afforded, in the hopes that this will find the global minimum.

We propose a modification of this classical SA algorithm, in which the cooling rate varies with the progress of the optimization, $k(T)$. The annealing schedule becomes a complicated function of the history of the past cooling rates,

$$T(t) \quad = \quad T_i e^{-\int k(T(t))\,\mathrm{d}t}, \tag{3.2}$$

but the actual cooling can be implemented quite easily using finite-difference decrements in the temperature using the instantaneous cooling rate,

$$T(t+\Delta t) \quad = \quad T(t) - k(T)\Delta t. \tag{3.3}$$

In principle, the cooling schedule $k(T)$ could be an arbitrarily complicated function; the optimal cooling schedule would be different for every system, and difficult to obtain. But the main intent of the variable cooling rate is to have the optimization proceed slowly only as the system cools across important transition temperatures, while cooling more rapidly away from these temperatures. Consequently, we propose a dual cooling rate approach, in which the cooling occurs at a fixed, slow rate, k_s, when the instantaneous heat capacity of the system is above some cutoff, C_V^*, and a different, faster rate, k_f, when the heat capacity

is below the cutoff:

$$k(T) \quad = \quad \begin{cases} k_s, & C_V(T) \geq C_V^* \\ k_f, & C_V(T) < C_V^* \end{cases} \tag{3.4}$$

To evaluate the heat capacity with no a priori information about the system, we make use of the fluctuation formula,

$$C_V \quad = \quad \frac{\langle E^2 \rangle - \langle E \rangle^2}{k_B T^2}, \tag{3.5}$$

in the canonical ensemble, where $\langle E \rangle$ and $\langle E^2 \rangle$ are evaluated at a fixed T.

Thus, a full implementation of the dual-cooling rate simulated annealing (DRSA) algorithm involves the following steps:

1. Begin at temperature T_i.
2. Equilibrate the system by sampling N_{eq} steps in the canonical ensemble at the current temperature.
3. Sample N_{prod} steps in the canonical ensemble at the current temperature, and use these to evaluate C_V (Eq. 3.5) and the associated cooling rate (Eq. 3.4).
4. Cool the system for N_{cool} steps using the current cooling rate (Eq. 3.3).
5. End the simulation, if the temperature has fallen to T_f; otherwise return to step 2.

We will refer to this modification of the SA algorithm as adaptive-cooling simulated annealing, or ACSA. Note that the method is characterized by eight different parameters: T_i, T_f, k_s, k_f, C_V^*, N_{eq}, N_{prod}, and N_{cool}. (The classical SA method requires only T_i, T_f, and k.)

Note that the requirement to evaluate the heat capacity at a fixed temperature builds some inefficiency into the optimization algorithm. Only some of the total sampling steps are used to cool the system; this productive fraction of the simulation is

$$f \quad = \quad \frac{N_{\text{cool}}}{N_{\text{eq}} + N_{\text{prod}} + N_{\text{cool}}}. \tag{3.6}$$

The remaining portion of the steps $(1 - f)$ represent the computational overhead required to evaluate the heat capacity using Eq. (3.5). The expectation is that the faster cooling rate k_f applied during some portions of the optimization will more than compensate for this overhead.

To measure the actual performance of the ACSA algorithm, and compare it to classical SA, we measure both the computational cost of the optimization as well as its accuracy. The computational cost, N, is the total number of sampling steps taken during the optimization. For classical SA, this can be determined directly from the cooling rate, along with the starting and ending temperatures: $N = \frac{1}{k\Delta t} \ln \frac{T_i}{T_f}$. For ACSA, it depends on the heat capacities calculated from the sampled states, and may be different for different trials. In practice, we measure the mean cost, $\langle N \rangle$, across many optimizations.

The accuracy of the optimization is determined by the probability, p, that an optimization finishes in the global-minimum energy basin. For this reason, all of the optimizations here are performed on LJ clusters for which the global minimum energy configuration is already known[40]. A configuration of an LJ cluster is defined to be in the global minimum basin if each of the interatomic distances $\{r_{ij}\}$ is identical to those in the global minimum energy configuration, to within a tolerance of $r_{\min} = \pm 0.24$ (reduced units). To estimate the accuracy of an optimization algorithm, it is run for n independent trials, using different pseudo-random number seeds. If n_{succ} of the trials terminated in the global minimum energy basin, and $n_{\text{fail}} = n - n_{\text{succ}}$ do not, then the success probability is estimated as $\langle p \rangle = n_{\text{succ}}/n$. Because this is a binomial process, the standard error of the measured mean accuracy is

$$\sigma_p = \sqrt{\frac{1}{n} p\,(1-p)}. \tag{3.7}$$

These two metrics, p and N, characterize the performance of an optimization method. Ideally, one would prefer a method with high p and low N. In practice, however, the accuracy is improved by reducing the cooling rate(s), which comes with increased computational cost. The important question, then, is how much computational effort is worth investing to improve the accuracy of the optimization method. This question has a natural answer when we observe that an inaccurate optimization method can be used to find the global optimum quite accurately when it is repeated in multiple independent trials.

For example, suppose an algorithm predicts a global minimum with some probability p. Even if p is not close to 1, so that there is a substantial probability that a single trial will fail to find the global minimum, there may still be a large probability that *most* of the trials will succeed. If we run this algorithm 3 times, independently, the probability that the method terminates in the global minimum more than half of the time (i.e. in 2 or 3 trials) is $p^3 + 3p^2(1-p) = p^2(3-2p)$. More generally, for n independent trials, the probability

that at least half of the trials succeed is

$$p_n = \sum_{k>n/2} \binom{n}{k} p^k (1-p)^{n-k}. \tag{3.8}$$

As long as $p > \frac{1}{2}$, the accuracy p_n increases monotonically with n. This technique is often used in practice to obtain useful results from an imperfect optimization method; if repeated optimizations identify the same final state, it can be declared the global minimum with more confidence than can the result of a single optimization.

Note that we intentionally require that the *majority* result match the global minimum, rather than the *lowest* energy obtained from multiple trials. This is because optimization algorithms are often applied to many-dimensional systems, which suffer from the curse of dimensionality. The number of states that are thermally accessible within an energy of kT_f above a local minimum scales as $T_f^{d/2}$ for a d-dimensional system. When a many-dimensional optimization is halted at a temperature T_f low enough to have localized to a particular energy basin, but above 0 K, it is extremely unlikely to be found at the true minimum. If there are local minima with energies within a few kT_f of the global minimum, then there is no guarantee that the lowest final energy is in the basin with the lowest minimum energy. If such distractor minima are not a problem, and it suffices to find the global energy basin a single time, then the n-trial success criterion in Eq. (3.8) can be replaced with

$$p_n = \sum_{k=1}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{3.9}$$

$$= 1-(1-p)^n. \tag{3.10}$$

This metric increases much more rapidly with n than does the majority-based p_n.

Note also that requiring a majority of the optimizations to find the global-minimum basin is overly conservative. If there are more than two local minima, and if the final states can be accurately clustered into distinct basins, then all that is needed is that a *plurality* of the final states be in the global energy basin. The accuracy metric in Eq. (3.8) could be replaced with its multinomial equivalent,

$$p_n = \sum_{k_0>k_j,\forall j} \binom{n}{k_0 k_1 \ldots} \prod_j p_{(j)}^{k_j} \tag{3.11}$$

However, this requires that the probabilities $p_{(j)}$ be known for each of the distractor minima, and Eq. (3.11) does not yield as easily to a continuous approximation, as discussed below, so we have not pursued this

approach further.

For a large number of trials, the cumulative distribution function in equation 3.8 is tedious to evaluate by direct summation. It is equivalent, and more convenient for large n, to evaluate p_n using the incomplete beta function. Assuming odd n,

$$\begin{aligned} p_n &= I_p\left(\frac{n+1}{2}, \frac{n+1}{2}\right) \\ &= \frac{\int_0^p t^{\frac{n-1}{2}} (1-t)^{\frac{n-1}{2}} t}{\int_0^1 t^{\frac{n-1}{2}} (1-t)^{\frac{n-1}{2}} t} \end{aligned} \tag{3.12}$$

Equation 3.8 or 3.12 describes the probability that an optimization, with single-trial success rate p, will arrive at the correct consensus result after n trials. An optimization approach with poor single-trial accuracy can be improved with repetition, and p_n describes exactly how fast the accuracy improves with repeated trials. This gives us the basis for defining a single scoring metric that combines accuracy and computational cost.

Consider a simplex optimization method with accuracy p and average computational effort N. To improve the accuracy, either the cooling rate can be reduced, or the optimizations can be repeated for multiple trials. Using n trials will improve the accuracy to p_n, at a cost of increasing the computational effort to nN. A modification to the optimization algorithm that delivers a higher accuracy of $p' = p_n$ in a single trial, should do so at a computational effort of less than nN, or else we would prefer repeated trials of the original optimization. In other words, an optimization method characterized by accuracy and effort (p, N) is equivalent to one characterized by (p_n, nN).

To quantify this relationship, let $\nu(p, p')$ be the number of times that a method with accuracy p would need to be repeated in order to achieve accuracy p'. That is, $p_{\nu(p,p')} = p'$.

This definition assumes $p \leq p'$. In cases where $p > p'$, the definition is extended so that

$$\nu(a,b) = \frac{1}{\nu(b,a)}. \tag{3.13}$$

In this way, ν can be interpreted as a real-valued measure of the relative cost of achieving two different accuracies (through repeated trials). (Note that although the number of repeated trials in Eq. 3.8 must be an integer, n has been extended to real values in Eq. 3.12.) For example, $\nu(0.90, 0.972) = 3$ because three repeated trials with accuracy 0.9 achieve a consensus accuracy of 0.972.

The efficiency of two methods with different accuracy and cost can be ranked by comparing the cost for each to obtain some benchmark accuracy. The computational effort required to achieve a target accuracy α, for a method that has single-trial accuracy p with computational effort N is

$$q_\alpha(p,N) \quad = \quad \nu(\alpha,p)N. \tag{3.14}$$

We use this normalized computational effort $q(p,N)$ as a scoring function to compare different optimization methods, usually with $\alpha = 0.9$. A low value of q corresponds to a more efficient optimization algorithm.

The parameters of the cooling method (3 for classical SA; 8 for ACSA) fully determine the normalized effort q for a particular system. This is determined by performing n independent trial optimizations to obtain $\langle N \rangle$ and $\langle p \rangle$, then using these to evaluate $q_\alpha\left(\langle p \rangle, \langle N \rangle\right)$.

In order to make a fair comparison between the new ACSA algorithm and the classical SA, the parameters of both methods were optimized to ensure they were as efficient as possible. The Nelder-Mead downhill simplex method [21] was used to optimize the parameters, using $n = 900$ trials for each point in parameter space to evaluate $\langle p \rangle$ and $\langle N \rangle$.

Once fully optimized parameters have been determined for both the classical SA and ACSA method on the same system, the efficiency of the ACSA method is defined using the ratio of their normalized computational efforts:

$$\varepsilon \quad = \quad \frac{q^{(\mathrm{SA})}}{q^{(\mathrm{ACSA})}}. \tag{3.15}$$

When ACSA can achieve the target accuracy with less computational effort than classical SA, the efficiency is greater than 1.

Accurate statistical estimation of q requires multiple, independent optimizations to be performed. Consequently, our implementation runs multiple simulated annealing optimizations in parallel. These parallel simulations are controlled, and their results are combined, using the message passing interface (MPI)[41] communications protocol, allowing high-throughput calculations even with large n. This communication is illustrated schematically in Fig. 3.1

The large-scale parallel replication of the optimizations is only needed here to perform a detailed analysis of the optimization method. Production-level application of the ACSA method would not require $n = 900$ repeated optimizations. However, a smaller number of repeated optimizations may be called for, in

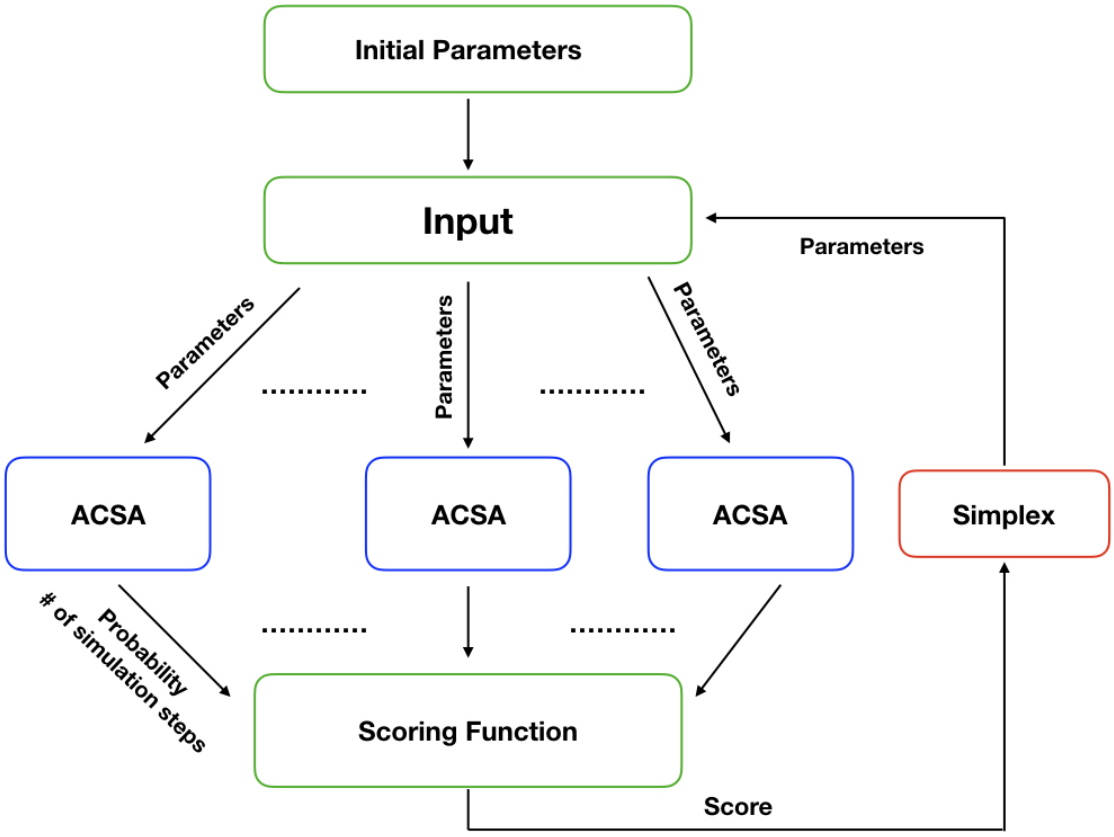

Figure 3.1: Parallel communication between simplex, adaptive-cooling simulated annealing, and the scoring function using MPI.

cases where the optimal set of parameters is one that generates a quick-and-dirty optimization with relatively low p, and counts on achieving additional accuracy through repeated trials.

The parameter optimization starts by running n ACSA optimizations at each vertex of an initial simplex in parameter space. After each ACSA simulation has finished, the mean success probability and the total computational cost at each vertex is used to evaluate q. This score is sent back to the simplex code, which uses it as the objective function value at that vertex. Once values are available at each vertex, the simplex algorithm has enough information to generate a new vertex, which requires a new ACSA optimization at the new vertex. The simplex algorithm continues with simplex moves until the calculated tolerance value (eq. 2.15) is lower than a tolerance value of 0.001. The vertex with the lowest q values is then taken to provide the optimal ACSA parameters. This method could be used for optimizing the classical SA parameters, as well. Classical simulated annealing requires 3 parameters: T_i, T_f, and k, but only T_f and k were optimized. Therefore, these parameters were originally optimized by a hand-directed grid search.

3.3 Preliminary results

We have applied the ACSA method to simple test cases, like one- and two-dimensional model functions. As an illustrative example, the results will be discussed in detail for a periodic two-dimensional potential with two wells, illustrated in Fig 3.2

Simulated annealing was performed using canonical-ensemble molecular dynamics to sample states on the potential energy function. The molecular dynamics was performed using the Langevin thermostat, using friction coefficient 0.5, and a velocity Verlet integrator with timestep of 0.002. The fictional particle on this potential had mass 1.0. The friction coefficient was chosen so that it was not so high that the particle spends too much time bouncing in one well and taking too much time to cross the energy barriers, and not so low that it exhibits too many barrier crossings and takes more time to thermalize into a particular basin. The time step was chosen to avoid energy conservation errors, which would require slower cooling. Very small time steps would result in a much longer simulation and would reduce the efficiency.

The parameters of classical SA were fully optimized, which correspond to the optimal value for the cooling rate k, and final temperature T_f. These optimal parameters result in the lowest score, which was calculated with the scoring function (eq. 3.14). This score can then be compared to the ACSA and used to compute the efficiency.

From our results we see that for faster cooling rates the probability for the system to end up in the

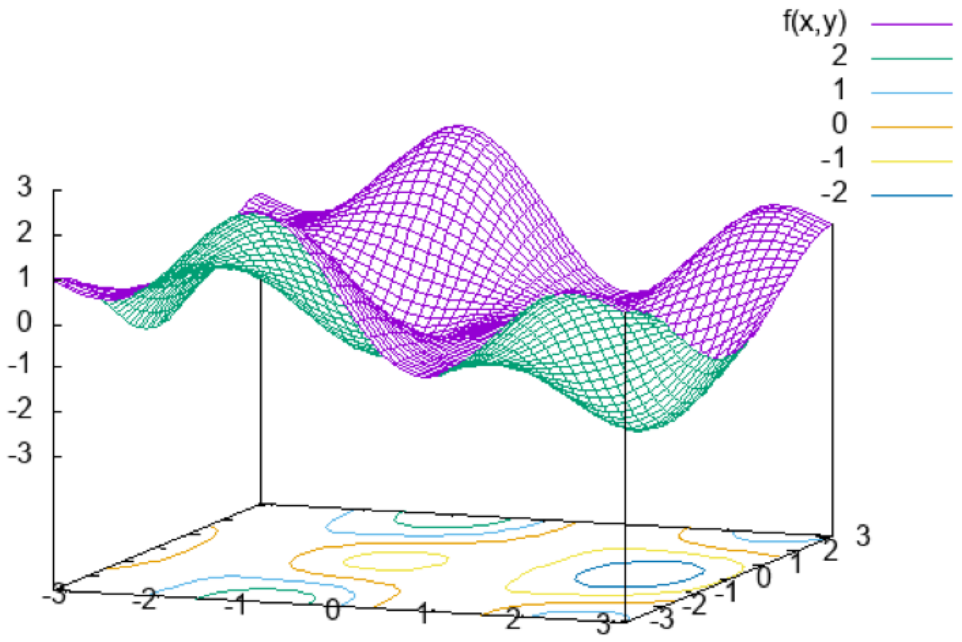

Figure 3.2: Two-dimensional two-well potential with $\Delta E = 5.5$: $f(x,y) = \sin 2x \cos y - \sin x - \cos y$

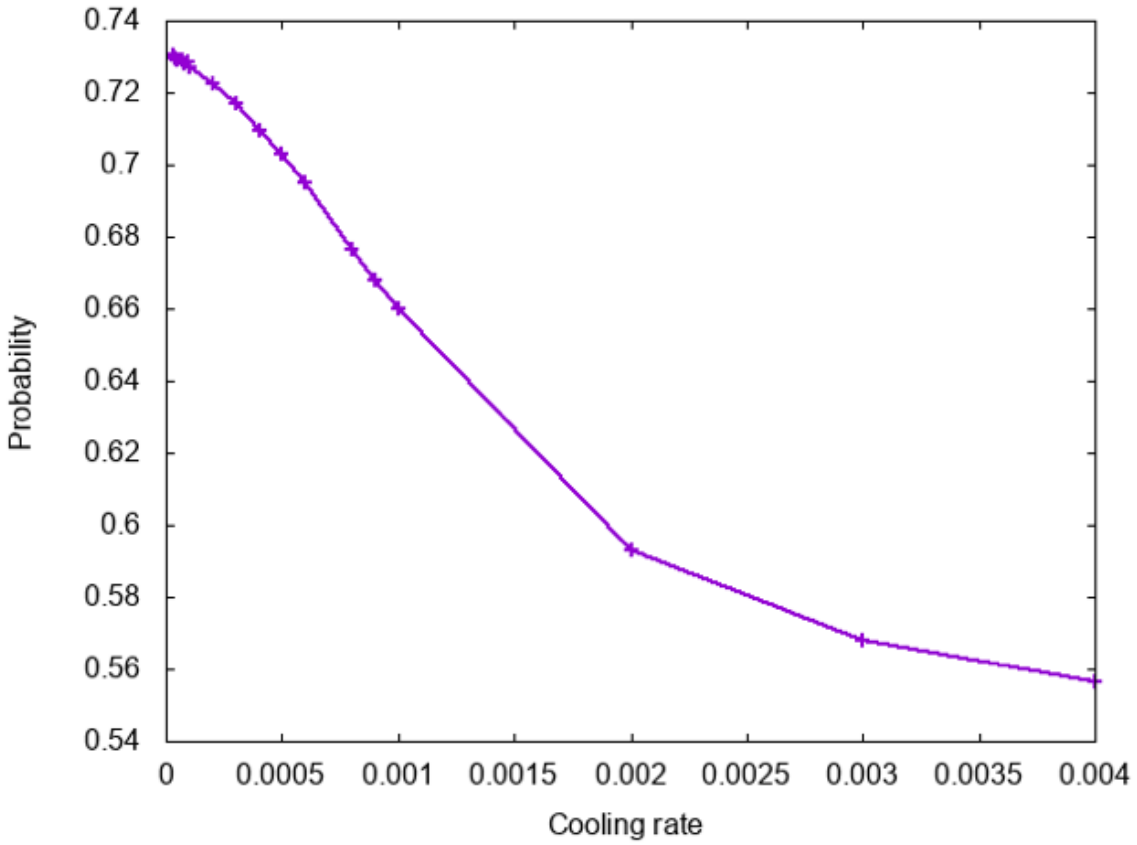

Figure 3.3: Probability of converging to the global minimum versus cooling rate k for the double-well 2-D objective function, optimized with classical SA.

global minimum decreases (Fig 3.3), as does the total number of the simulation steps (Fig. 3.4).

A minimal final score for the optimization is achieved at some optimal cooling rate; this score is calculated from the probability and total number of simulation steps, as described in the previous section and equation 3.14. This is the score that we will use to compare the efficiency of the classical SA to our ACSA. (Fig. 3.5)

From these results we can conclude that the optimal cooling rate for this objective function is close to 0.001, which results in a minimal score. Note that the probability of a successful optimization at this cooling rate is 0.66022 ± 0.00066, which may seem low for an "optimal" optimization. However, achieving higher accuracy begins to require significantly longer simulations, as indicated in Figure 3.4. As the probability decreases at the faster cooling rates, we would need to run the simulation many more times to reach the desired probability of 0.90. Therefore, the final score for the simulation increases. With slower cooling, on the other hand, we get higher probability of success and we need to run the simulation fewer times, but the

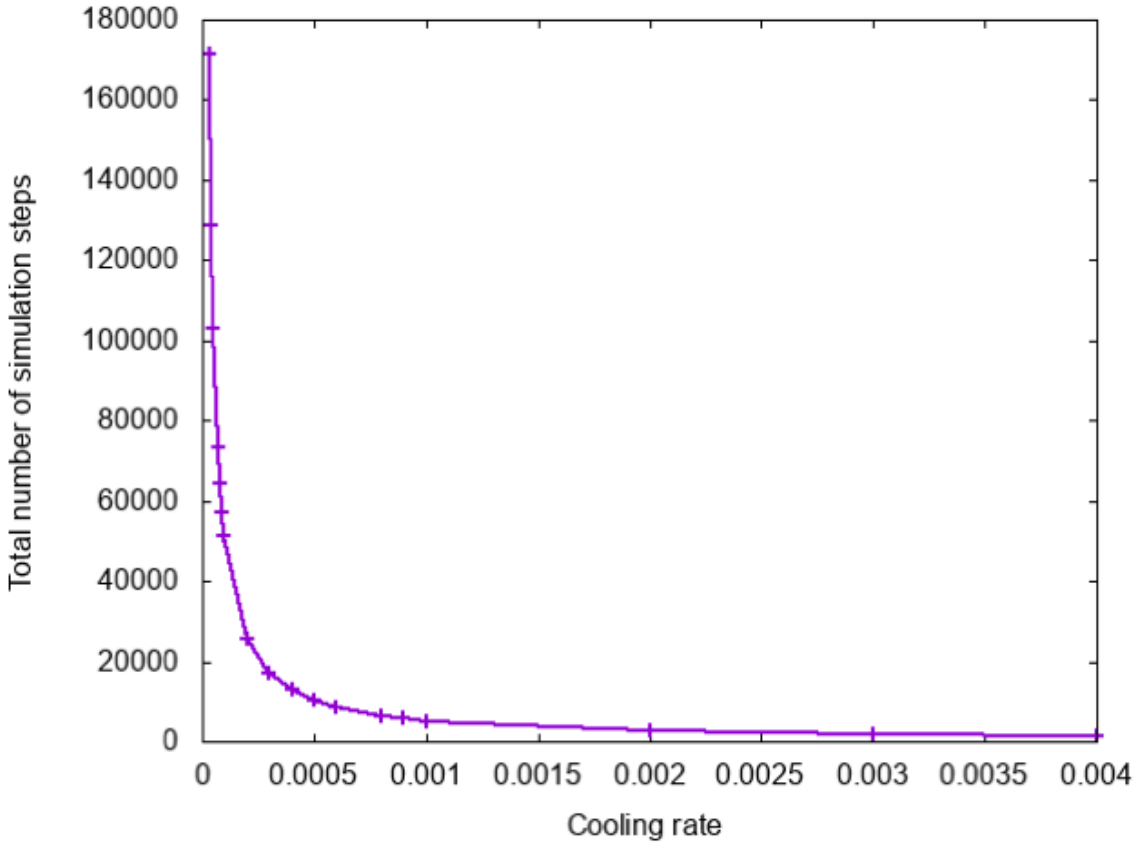

Figure 3.4: Total number of simulation steps versus cooling rate for the two-well two- dimensional objective function

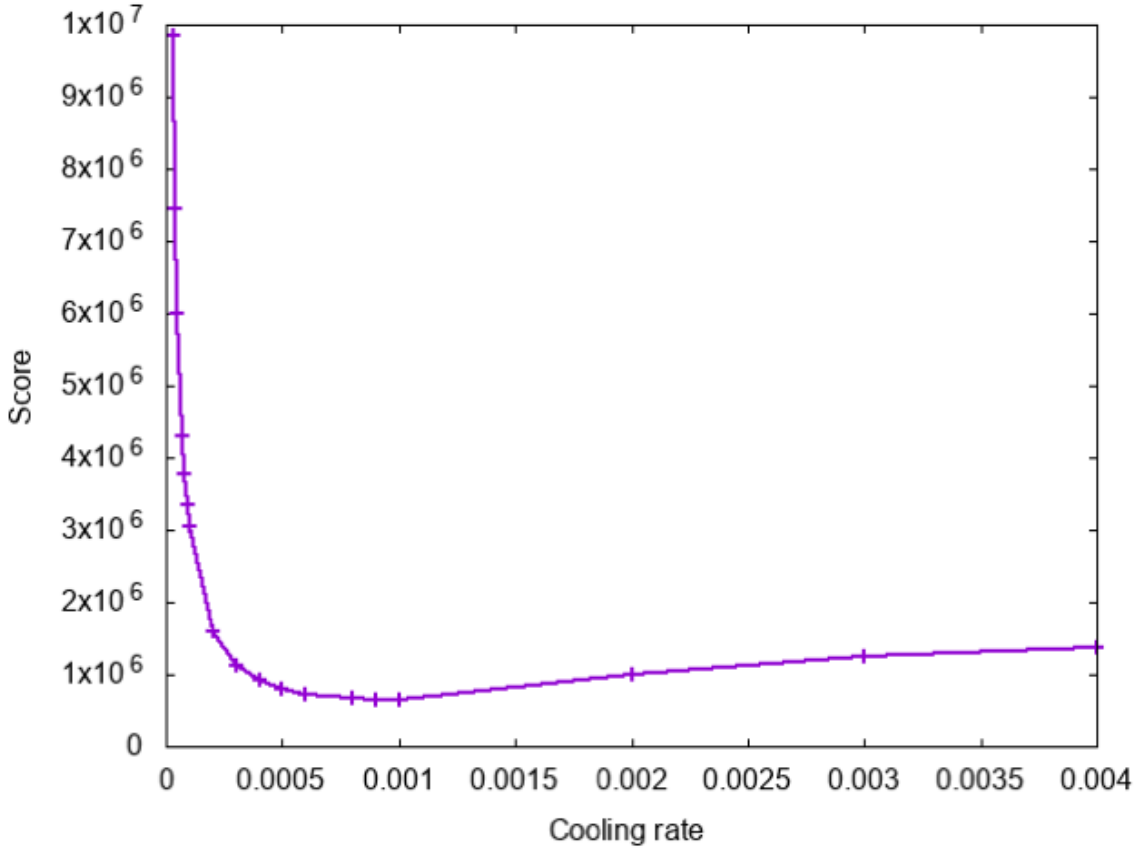

Figure 3.5: Score versus cooling rate for the two-well two-dimensional objective function, optimized with classical SA, and calculated with the use of our scoring function.

total number of simulation steps increases, which results in a higher score. Therefore, there is an optimal cooling rate, which gives us a minimal score even when the probability is lower than the target accuracy of 0.90.

ACSA depends heavily on the heat capacity that is calculated on the fly during the simulation as well as on the heat capacity cutoff, which is optimized by simplex. Our algorithm chooses a slow cooling rate if the instantaneous value of the heat capacity is higher than the heat capacity cutoff. Therefore, if our heat capacity computation is too noisy, then the wrong cooling rate might be chosen more often. For instance, if the true value of the heat capacity is below the heat capacity cutoff, but due to the stochastic noise sampling the instantaneous value of the heat capacity is above the cutoff, then the slow cooling rate will be chosen more often. This means that the system will be cooled down at the slow cooling mode too much time and we will not be able to get a much better efficiency comparing to the classical SA due to the longer optimization time. On the other hand, if the true value of the heat capacity is above the heat capacity cutoff, but due to the noise the instantaneous values of the heat capacity are below the cutoff, then the fast cooling rate might be chosen too often, which might decrease the efficiency of the method by decreasing the probability of converging to the global minimum.

Ideally, the heat capacity versus temperature graph should look like the one in Fig. 3.6, which is based on the theoretical calculation of the heat capacity using equation 3.5. To evaluate the mean energy and mean square energy from equation 3.5 theoretically, numerical integration was used with a grid-based approach. The heat capacity peak is expected at 5500 temperature units, when just a slight change in temperature allows the system to explore significantly more local states.

From a graph of the heat capacity as a function of simulation temperature during a ACSA (Fig. 3.7), we can see that the heat capacity graph is a bit noisy, but the simulation still spends the majority of its cycles using the fast cooling rate, which decreases the optimization time. Slow cooling is used for only a minority of the simulation cycles. The highest heat capacity is observed at 18000 temperature units. On both Fig. 3.6 and Fig. 3.7 we do observe a heat capacity peak, though results from ACSA (Fig. 3.7) are noisier than the theoretical result in Fig. 3.6. Since the heat capacity is calculated based on energy fluctuations (eq. 3.5) and with a finite amount of sampling, it contains some statistical noise. Also, from Fig. 3.7 we can see that ACSA is using a slow cooling rate in the proper temperature region, where the true heat capacity peak is expected, from 18000 to 3500 temperature units.

ACSA parameters were optimized with simplex (Table 3.1). From Table 3.1 we can see that both cooling rates for ACSA are $\sim$3 times faster than the slow cooling rate in classical SA. The final score for

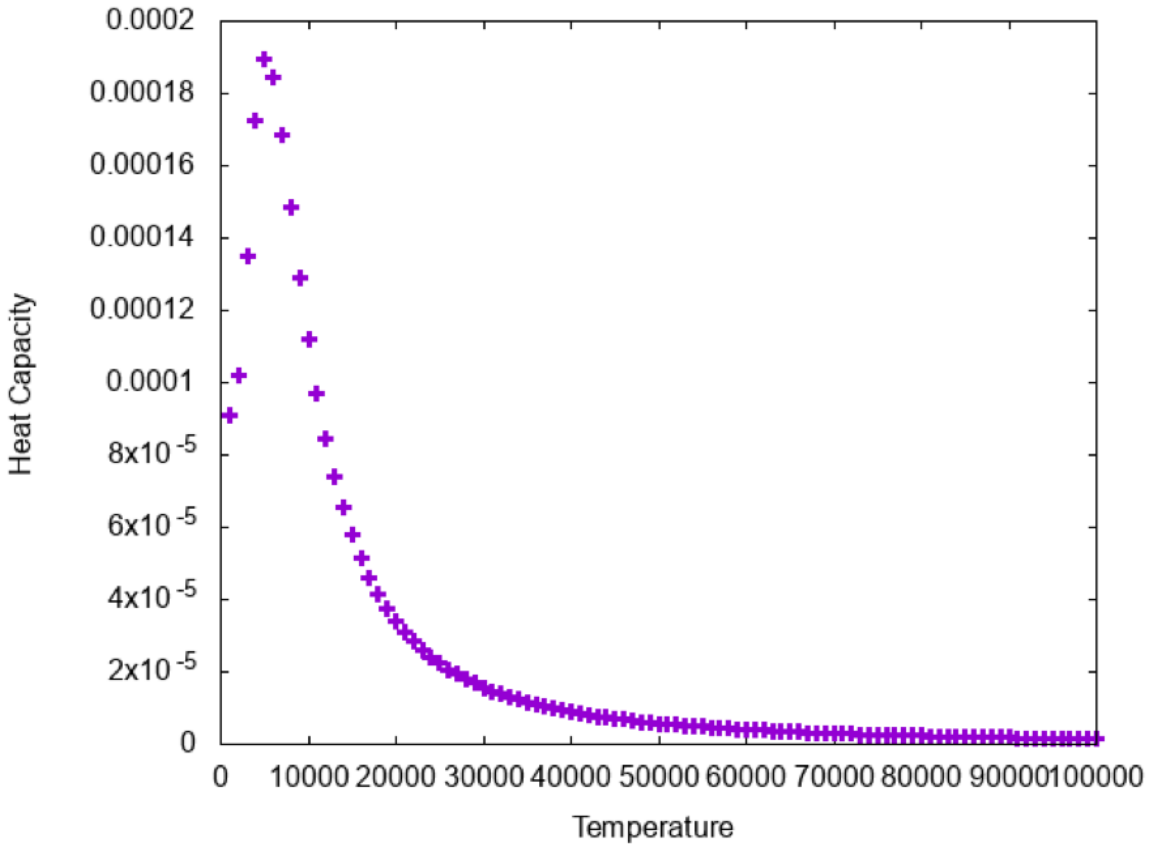

Figure 3.6: Heat capacity versus temperature for the two-well two-dimensional objective function, calculated using direct evaluation of equation 3.5

Table 3.1: Final parameters and scores for two-dimensional two-well function optimized by ACSA and classical SA.

Method	k (k_s)	k_f	$\langle p \rangle$	$\langle N \rangle$	q
ACSA	3.00×10^{-3}	3.72×10^{-3}	0.6913	6.99×10^{3}	5.92×10^{5}
Classical SA	1.00×10^{-3}	NA	0.6602	5.14×10^{3}	6.55×10^{5}

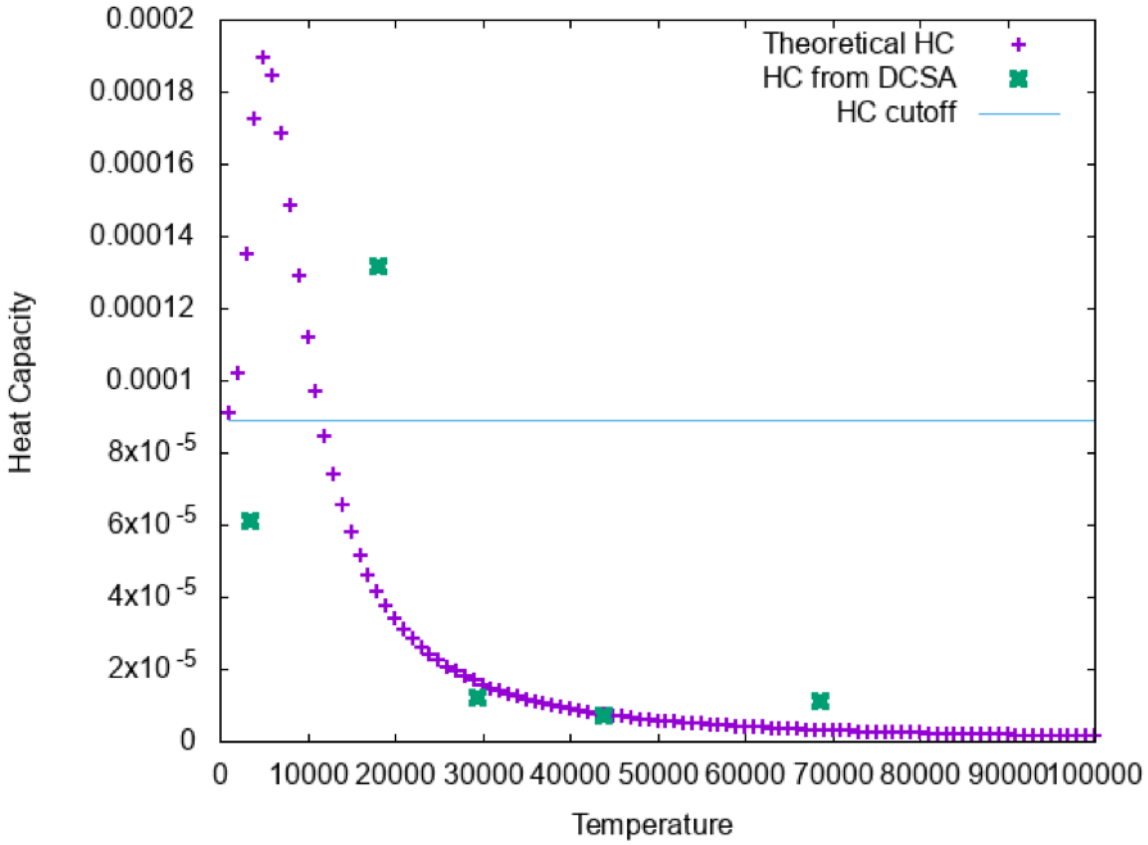

Figure 3.7: Heat capacity versus temperature graph for the two-well two- dimensional objective function, optimized by ACSA and simplex. Number of equilibration steps N_{eq} = 517, number of production steps N_{prod} = 417, number of cooling steps N_{cool} = 276.

Table 3.2: Efficiency of the ACSA for different objective functions.

Dimensionality	Number of minima	Efficiency
1D	3	1.039
1D	7	14.274
2D	2	1.106
2D	3	7.875

ACSA is lower than in classical SA because the final probability of ending up in a global minimum is higher in ACSA. Ideally, a slower cooling rate should result in a higher probability. Therefore, it can be concluded that ACSA includes some statistical noise due to the finite amount of sampling, since it results in a higher probability with faster cooling rates than the classical SA. For this objective function, the ACSA method was 1.106 times more efficient than the classical SA (Table 3.2). The efficiency was calculated by dividing the minimal score of the fully optimized objective function of the classical SA by the minimal score of the same objective function fully optimized by ACSA.

We have tested a series of model potential energy surfaces (Fig. 3.8 – 3.10), which differ in dimensionality and the number of wells, and our goal was to see how our method would do when the complexity of the objective function increases (Table 3.2). All of the objective functions were designed from linear combinations of *sin* and *cos* terms, and were periodic over some domain. For one-dimensional function with three wells the efficiency of the ACSA over classical SA is 1.039, which is almost the same as classical SA. But adding more dimensions and wells increases the efficiency. For example, the efficiency of ACSA over classical SA for a two-dimensional function with three wells is 7.875 which is much higher when comparing to the efficiency of 1.039 for one-dimensional function with three wells. Therefore, we observe that the efficiency of the ACSA appears to increase as the complexity of the objective function increases (Table 3.2). The highest efficiency of 14.274 was achieved for a one-dimensional seven-well function.

In general, this can be explained by the fact that classical simulated annealing requires a slower cooling rate as the complexity of the objective function increases, and this leads also to the longer optimization time. From tables 3.1 and 3.3 we can see that the two- dimensional three-well function (Table 3.3) requires 25 times slower cooling than the two-dimensional two-well function (Table 3.1). In case with two-dimensional three-well function, fast cooling rate in ACSA, is 25 times faster than the cooling rate used in classical SA, due to which the efficiency of the ACSA over the classical SA drastically increases. We believe that the ACSA method would be even more efficient for more complex chemical/atomistic systems like proteins, Lennard-Jones clusters and polymers.

Table 3.3: Final parameters and scores for two-dimensional two-well function optimized by ACSA and classical SA.

Method	k (k_s)	k_f	$\langle p \rangle$	$\langle N \rangle$	q
ACSA	7.50×10^{-5}	1.07×10^{-3}	0.4826	1.59×10^{4}	7.37×10^{6}
Classical SA	4.00×10^{-5}	NA	0.5744	1.21×10^{5}	7.48×10^{7}

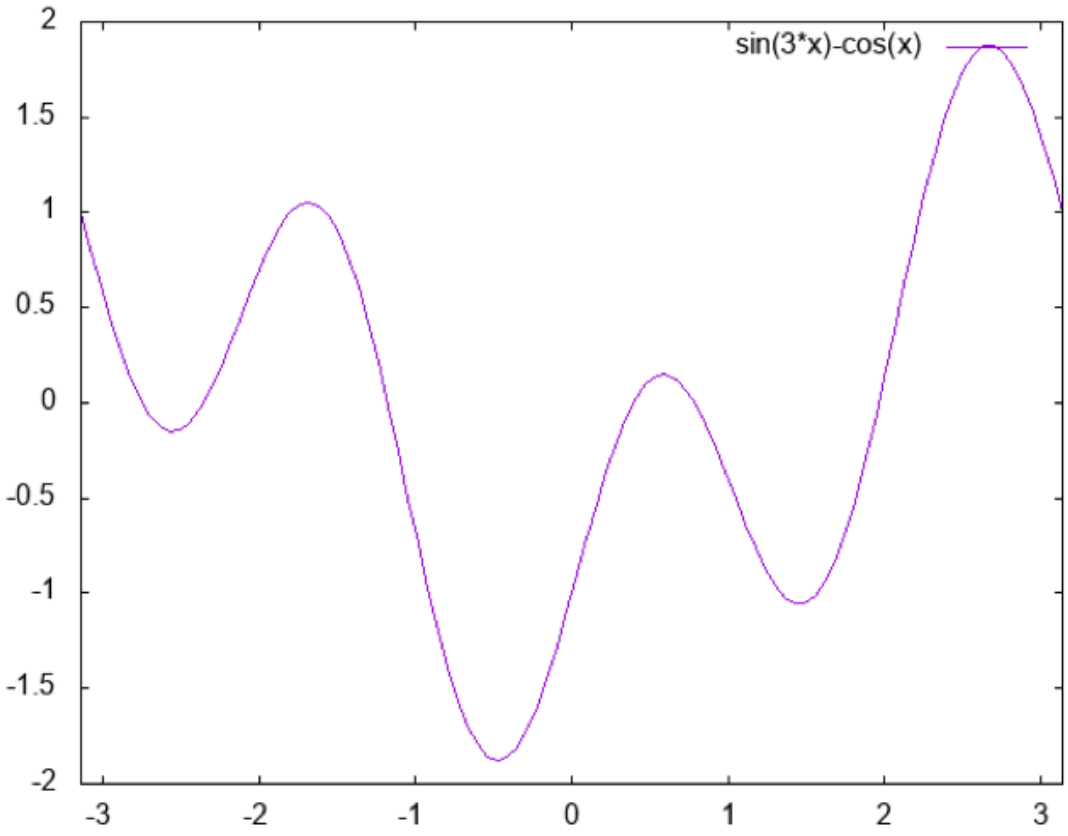

Figure 3.8: One-dimensional three-well potential function: f(x) = $\sin 3x - \cos x$

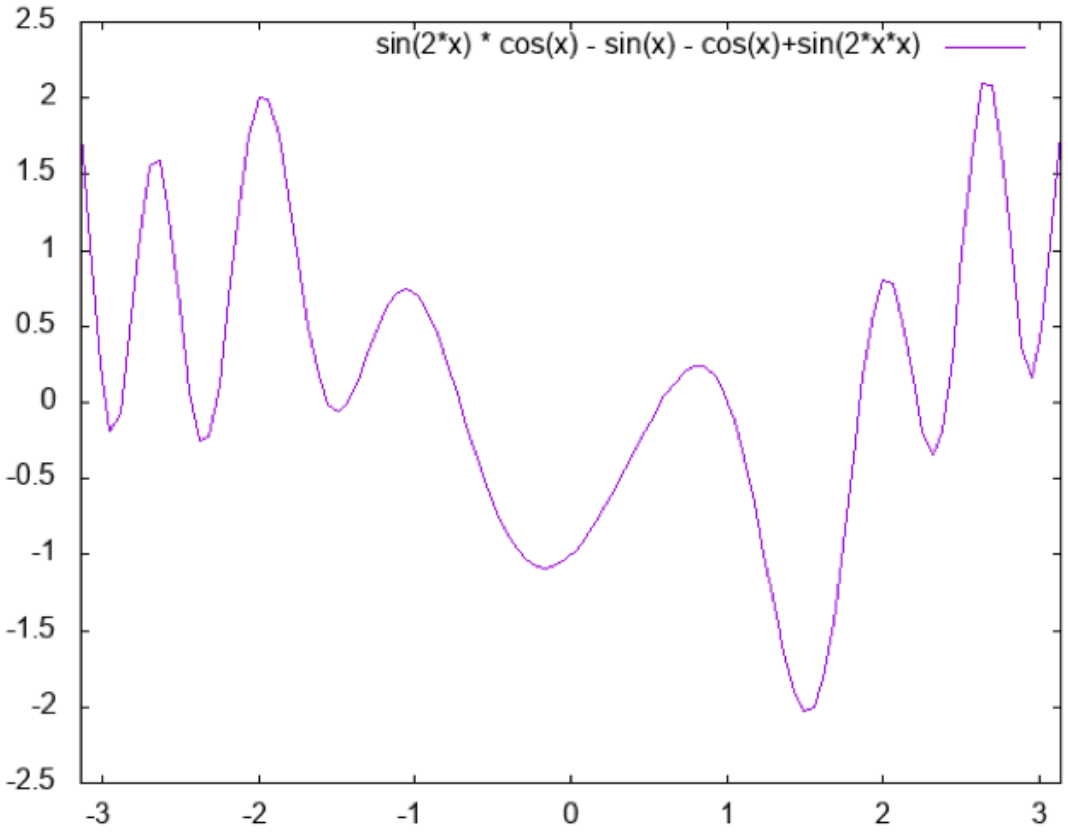

Figure 3.9: One-dimensional seven-well potential function: $f(x) = \sin 2x \cos x - \sin x - \cos x + \sin 2x^2$

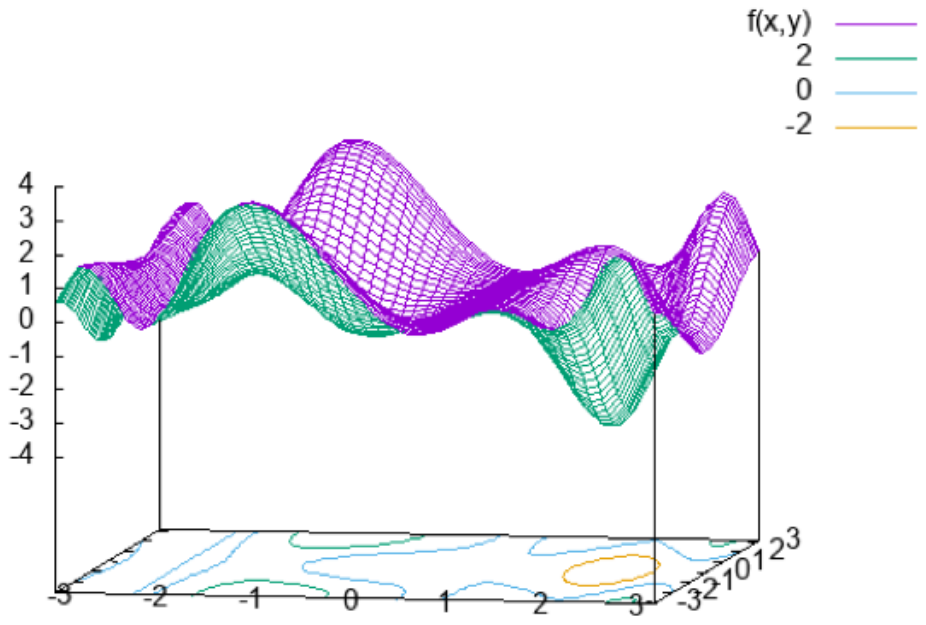

Figure 3.10: Two-dimensional three-well potential function: f(x,y) = $\sin 2x \cos y$ - $\sin x$ - $\cos y$ + $\sin x^2$

Table 3.4: Optimized parameters for classical SA applied to energy minimization of LJ_n clusters.

n	k	T_i[1]	T_f
6	4.50×10^{-6}	0.150	0.0020
7	5.39×10^{-7}	0.250	0.0354
9	6.50×10^{-7}	0.220	0.0395
10	3.73×10^{-7}	0.260	0.0869
13	6.16×10^{-7}	0.310	0.0867
19	3.52×10^{-7}	0.315	0.1299
20	7.28×10^{-7}	0.380	0.1087
23	9.38×10^{-7}	0.400	0.1385
36	1.15×10^{-6}	0.190	0.1180

3.4 Lennard-Jones clusters

To test the efficiency of the ACSA method, energy minimizations were performed on a number of Lennard-Jones (LJ) clusters, LJ_n. This system has been well studied as a benchmark for optimization methods, and the minimum-energy structures are known[40] for clusters with n up to at least 1000. Results will be discussed in detail for clusters with $n = 6, 7, 9, 10, 13, 19, 20, 23, 36$. These clusters were chosen to span a range of cluster sizes and complexities, while still being small enough to explore with statistical thoroughness.

Initial geometries of the LJ_n clusters were obtained by placing the atoms randomly, with uniform distribution, within a distance of 2.74σ of the origin. The sampling steps of the simulated annealing algorithm were performed using molecular dynamics with the velocity Verlet thermostat[11], using a timestep of 0.002 in reduced LJ units. Temperature control during the equilibration and production phases was achieved using the Langevin thermostat[11], with a friction coefficient of 0.002, also in reduced LJ units.

Because the heat capacities of LJ_n clusters are well understood as a function of temperature,[42, 43] the initial temperature for both the SA and ACSA optimizations was chosen to be a value that was somewhat above the peak in heat capacity. All other parameters (7 for ACSA, 2 for SA) were optimized to minimize the value of $q_{0.9}$. The optimized parameters for both methods are shown in Tables 3.4 and 3.5. The performance of the two methods with these optimized parameters is compared in Table 3.6.

Fig. 3.11 shows the efficiency of the ACSA algorithm, relative to classical SA, and how this efficiency depends on cluster size. Although the methods are comparable in efficiency at small cluster size, the ACSA algorithm begins to perform better as the cluster size increases. In particular, the ACSA algorithm is able to find the minimum-energy cluster more than twice as fast as classical SA (even after accounting for

[1] Initial temperature was set, not optimized.

Table 3.5: Optimized parameters for ACSA applied to energy minimization of LJ_n clusters.

n	k_s	k_f	C_V^*	T_i	T_f	N_{eq}	N_{prod}	N_{cool}
6	4.74×10^{-6}	$1.35 \cdot 10^{-5}$	8.61	0.150	0.0015	91	75	96
7	4.21×10^{-7}	1.06×10^{-5}	3.74	0.250	0.0175	79	81	70
9	1.30×10^{-6}	8.81×10^{-6}	3.80	0.220	0.0330	284	332	102
10	4.80×10^{-7}	1.91×10^{-5}	4.47	0.260	0.0326	78	94	70
13	6.88×10^{-7}	7.33×10^{-6}	3.33	0.310	0.0617	95	250	103
19	5.70×10^{-7}	1.85×10^{-5}	4.90	0.315	0.0723	30	47	117
20	8.64×10^{-7}	9.12×10^{-6}	5.83	0.380	0.0656	55	37	92
23	1.06×10^{-6}	1.39×10^{-4}	5.96	0.400	0.0466	59	62	125
36	1.04×10^{-6}	6.30×10^{-5}	4.35	0.190	0.1473	35	28	70

Table 3.6: Performance of the classical SA and ACSA algorithms for energy minimization of LJ_n clusters[2, 3].

		SA			ACSA			
n	N_{min}	$\langle p \rangle$	$\langle N \rangle$	q	$\langle p \rangle$	$\langle N \rangle$	q	ε
6	2	0.568	9.59×10^5	9.59×10^6	0.540	8.61×10^5	1.00×10^7	0.95
7	4	0.9056	3.63×10^6	1.08×10^7	0.799	1.59×10^6	9.34×10^6	1.16
9	21	0.9556	2.64×10^6	5.28×10^6	0.9567	1.49×10^6	4.37×10^6	1.21
10	64	0.9144	2.94×10^6	8.82×10^6	0.693	1.22×10^6	9.03×10^6	0.98
13	1510	0.9067	2.07×10^6	6.20×10^6	0.9100	1.35×10^6	5.00×10^6	1.24
19	$\sim 2 \times 10^6$	0.706	2.51×10^6	1.50×10^7	0.698	1.29×10^6	9.53×10^6	1.55
20		0.712	1.72×10^6	1.03×10^7	0.793	1.50×10^6	8.79×10^6	1.17
23		0.268	1.13×10^6	2.60×10^7	0.550	1.03×10^6	1.21×10^7	2.15
36		0.158	5.80×10^5	1.94×10^7	0.257	2.46×10^5	7.29×10^6	2.67

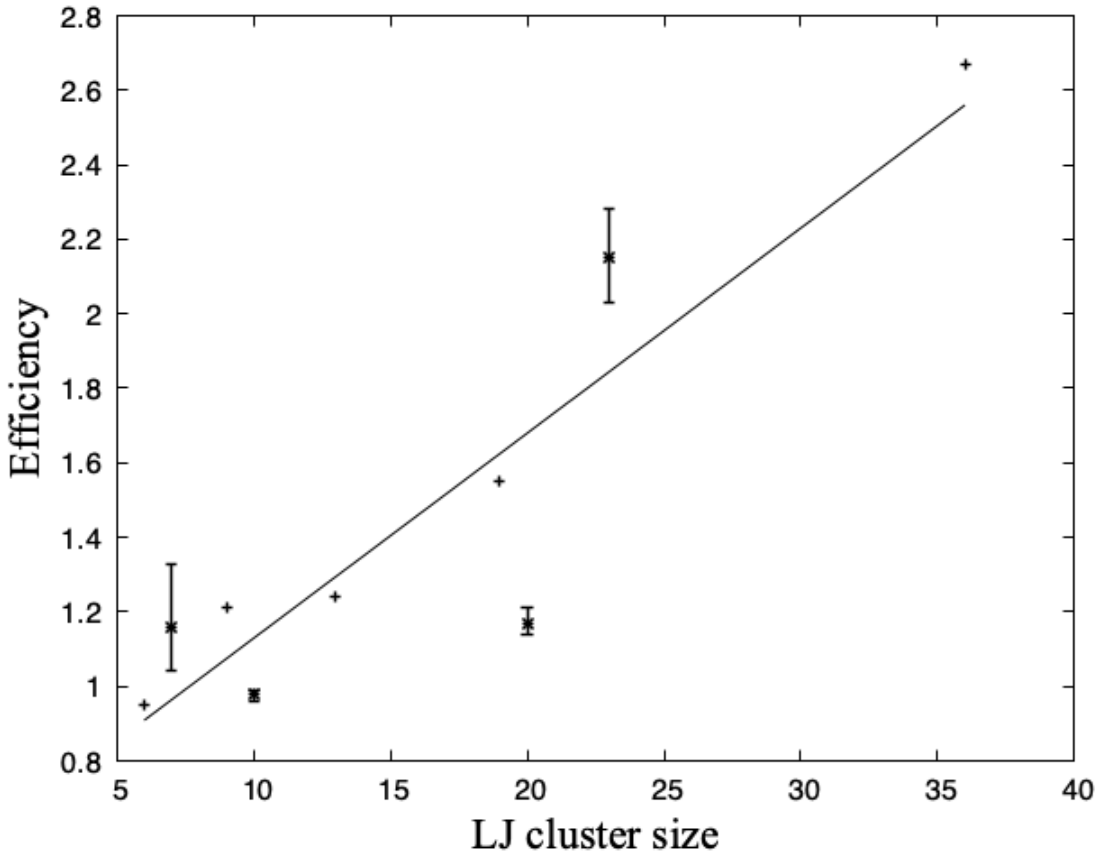

Figure 3.11: Efficiency of the ACSA over the classical SA for LJ clusters. Error bars represent the standard error of the mean. The straight line is a least squares fit, but is only intended as a guide to the eye.

the overhead in evaluating the heat capacity), once the cluster size exceeds $n = 20$. It is reasonable to expect that this advantage will continue to increase for larger, more computationally demanding optimizations.

It is interesting to explore the reasons for the computational advantage of the ACSA algorithm. Fig. 3.12 compares the ACSA slow cooling rate, k_s, to the optimal SA cooling rate at each cluster size. These cooling rates are quite similar, thus confirming the initial motivation for the method: The classical SA algorithm performs most efficiently with a cooling rate that provides a balance between low error rates and fast optimization; the ACSA algorithm independently uses very nearly the same optimal cooling rate, but only in the crucial region of phase space where the number of thermally accessibly states is decreasing rapidly. This point is emphasized further in Fig. 3.13, which shows that the ratio $k_s(\mathrm{SA})/k(\mathrm{ACSA})$ is always relatively close to 1, even though the magnitude of the individual k values varies by as much as a factor of 10 for different clusters.

The classical SA method cools at the same rate as it anneals through all regions of phase space, even

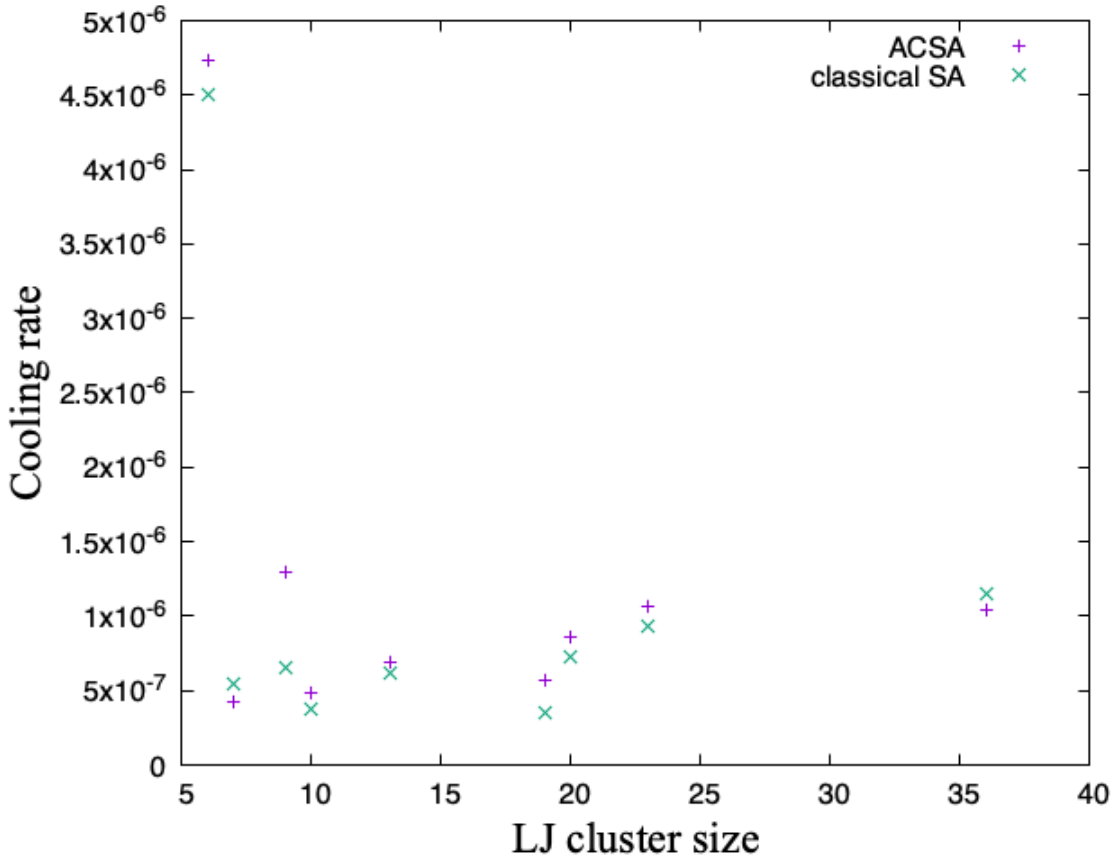

Figure 3.12: Optimal classical SA cooling rates (k) and ACSA slow cooling rates (k_s) for a number of LJ cluster sizes.

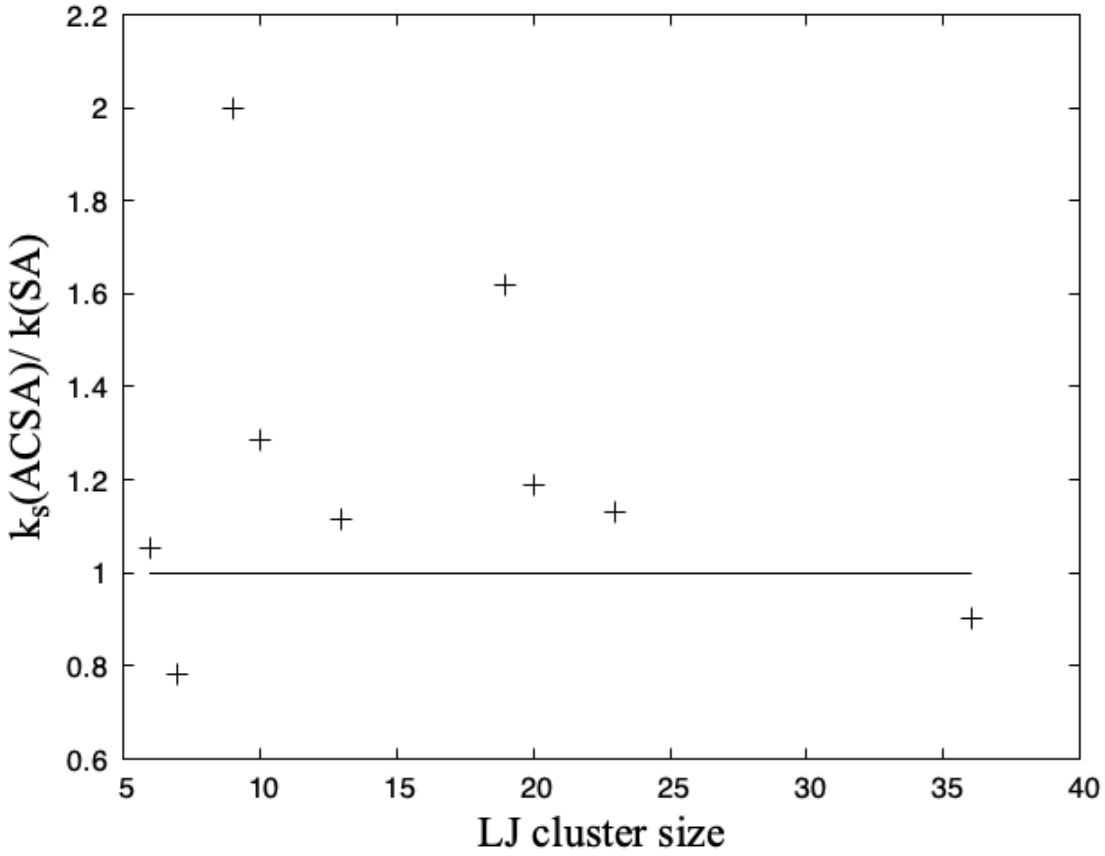

Figure 3.13: Slow cooling rates for ACSA and classical SA for LJ clusters.

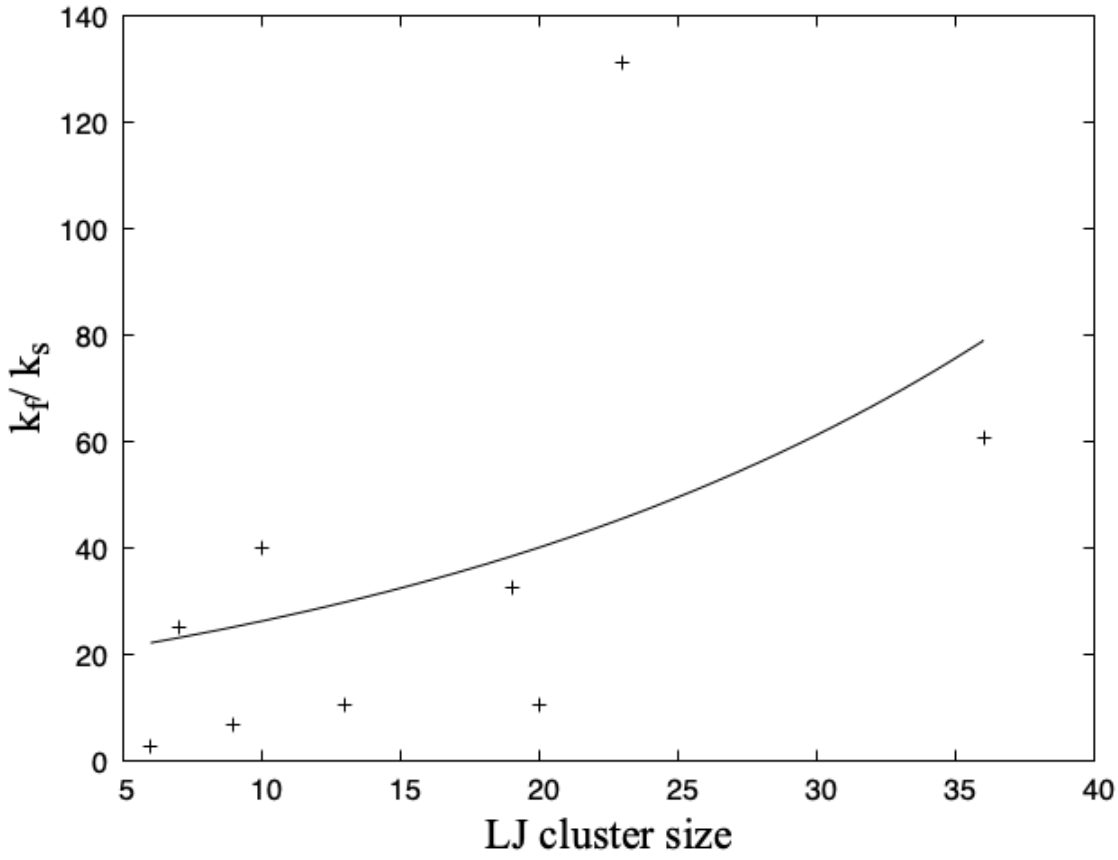

Figure 3.14: Ratio of the fast cooling rate to the slow cooling in ACSA for several cluster sizes.

those regions that have relatively low risk of quenching into local minima. The ACSA algorithm, on the other hand, has the flexibility to cool at a faster rate when this kinetic trapping risk is low. Fig. 3.14 shows the ratio k_f/k_s for the ACSA algorithm. The fast cooling rates are always at least several-fold faster than the slow cooling rates, even for the smallest clusters. This ratio increases with increasing cluster sizes, exceeding a factor of 50 for the larger clusters ($n > 20$) where ACSA is most efficient.

Thus, the speedup obtained by the ACSA algorithm results from these periods of faster cooling. The cooling is nearly the same as classical SA in the crucial bottleneck regions of the energy landscape, but much faster at other times. The computational advantage of this faster cooling is more than enough to make up for any increased error rate, as well as well as the computational overhead associated with evaluating the heat capacity.

This point is reinforced by examining how the values of $\langle p \rangle$ and $\langle N \rangle$ differ for ACSA from the values for classical SA (see Table 3.6). The optimal ACSA value of $\langle p \rangle$ is in some cases larger than the value

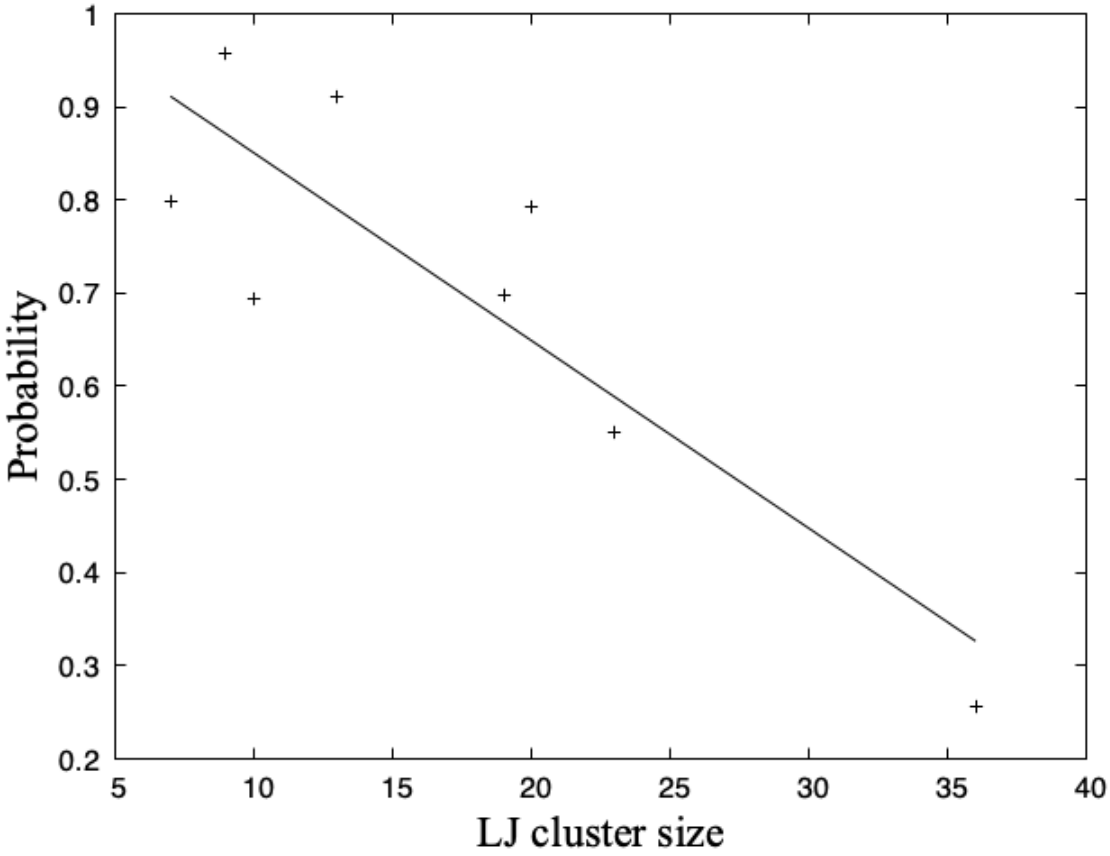

Figure 3.15: Probability of successfully reaching the global minimum as a function of LJ cluster size.

for SA, and in some cases smaller. But the ACSA algorithm always succeeds in finding the minimum more quickly, with a smaller value of $\langle N \rangle$. Regardless of whether the modified cooling rates result in a more or less accurate optimization, the benefit comes from reducing the effort required to reach the answer.

The (optimal) probability of minimizing into the correct global minimum decreases as the cluster size increases, as can be seen from both Table 3.6 and Fig. 3.15. This is not surprising, as it is due to the rapid increase in the number of local minima with increasing dimensionality of the energy landscape. (Table 3.6 lists the number of local minima[2, 3] for the smaller clusters. This value rises steeply enough that the energy landscape has not been fully explored for even moderately large clusters.) With the exception of the $n = 6$ cluster (which has only two local minima, and can tolerate an anomalously fast cooling rate and correspondingly poor success rate), most of the clusters have an increasingly hard time finding the global minimum as the cluster size increases.

Table 3.5 also lists the optimized number of equilibration, production, and cooling steps chosen for

the ACSA algorithm. There is considerable variation in these values, but as a general observation it appears that the number of steps in each block is roughly equal. Averaging over all cluster sizes, 30% of the steps are spent equilibrating, 38% in production evaluating the heat capacity, and 32% cooling the system. Thus, roughly a third of the computational effort is spent actually annealing the system, with about two thirds of the computational effort spent on the overhead required to evaluate the heat capacity.

One advantage of the classical SA algorithm over the new ACSA algorithm is its smaller number of parameters (1 cooling rate and 3 total parameters, vs 2 cooling rates and 8 total parameters). The data presented so far has been for uses of the algorithms (both SA and ACSA) with a set of parameters that has been fully optimized for each cluster size. The computational effort required to optimize these parameters, of course, is several orders of magnitude larger than the computational effort required for a single optimization. Thus, the computational advantage of the ACSA algorithm over classical SA will only be useful if it can be implemented without requiring a full parameter optimization. Thus, we also performed several tests to gauge the success of the ACSA algorithm on LJ clusters without the benefit of a full parameter optimization. In the first of these tests, the ACSA parameters were chosen based only on the values of the (optimal) classical SA parameters. In this test, the classical SA algorithm has the advantage of using optimized parameters, but the ACSA parameters are chosen using heuristics so as not to involve any additional computational effort. The previously studied LJ_{23} cluster was used for this test. In the second test, both the SA and ACSA parameters were chosen using heuristics, for a previously unstudied cluster. The LJ_{24} cluster was used in this case.

Several rules of thumb for determining ACSA parameters were suggested by the preceding results. Fig. 3.13 suggests the heuristic that

$$k_s = \alpha k, \tag{3.16}$$

with $\alpha = 1.23$. That is, the ACSA slow cooling rate is taken to be nearly the same as the SA cooling rate (slightly larger, based on the cases examined so far).

Similarly,

$$k_f = n^{\beta} k, \tag{3.17}$$

where n is the LJ cluster size and we take $\beta = 1.139$.

That is, the ACSA fast cooling rate increases with cluster size, as seen in Fig. 3.14. The initial

Table 3.7: Predicted (non-optimized) ACSA parameters for the LJ cluster optimizations.

n	k_s	k_f	C_V^*	T_i	T_f	N_{eq}	N_{prod}	N_{cool}
23	1.15×10^{-6}	3.33×10^{-5}	6.0	0.4	0.069	90	108	145
24	1.02×10^{-6}	6.5×10^{-5}	5.6	0.4	0.12	45	50	114
46	1.30×10^{-6}	1.56×10^{-4}	3.5	0.4	0.01	90	108	145

temperature is taken to be the same for both SA and ACSA ($T_i = 0.4$), for LJ_{23}. As with all clusters, this was taken to be a temperature somewhat above the observed peak in heat capacity.[42, 43] The final temperature for ACSA is taken to be half that of the final temperature for the classical SA, as a relatively conservative example of the behavior seen in Tables 3.7 and 3.4), where the optimal T_f for ACSA is lower than the optimal T_f for classical SA. Presumably this occurs because the ACSA algorithm is cooling at the fast rate in the late stages of the optimization, and thus does so with less computational cost, pushing the balance towards the slightly higher accuracy achieved by cooling more thoroughly. The heat capacity cutoff, C_V^*, is chosen by observing that the optimal value of C_V^* for the other clusters is roughly half of the peak heat capacity value for that LJ cluster.[42, 43] For the LJ_{23} cluster, this corresponds to a value of $C_V^* = 6.0$, in reduced units. Lastly, the number of steps in the equilibration, production, and cooling phases were chosen using

$$N_{Cool} \quad = \quad \gamma \cdot (N_{Eq} + N_{Prod}) \tag{3.18}$$

For the LJ_{23} cluster optimization we chose γ of 0.73 to predict the number of cooling optimization steps, which is a mean value γ for previously optimized clusters. On average, the simulations require about 1.2 times more production steps than equilibration simulation steps, and the ratio slowly decreases for the clusters larger than LJ_{13}. The number of equilibration steps for LJ_{23} is a mean value of the equilibration steps for previously optimized clusters.

The parameters predicted by these heuristics are summarized in Table 3.7, and the performance of the resulting optimization is summarized in Table 3.8. Even without optimized parameters, the ACSA algorithm is 1.06 times more efficient than the (fully optimized) classical SA algorithm. This is not as good as the efficiency of $\varepsilon = 2.15$ value achieved with optimized parameters, but it illustrates that the ACSA algorithm can outperform SA without any additional effort spent on tuning the ACSA performance.

It is more typically the case that the optimal SA parameters are not known either, and are estimated heuristically, based on trial and error, or past experience. For the LJ_{24} and LJ_{46} clusters, where the optimal parameters are not known, we estimated the parameters for both SA and ACSA, based on the previous optimizations.

Table 3.8: Performance of the classical SA and ACSA algorithms for the test cases with heuristically predicted parameters.

	SA			ACSA			
n	$\langle p \rangle$	$\langle N \rangle$	q	$\langle p \rangle$	$\langle N \rangle$	q	ε
23	0.268	1.13×10^6	2.60×10^7	0.438	1.75×10^6	2.45×10^7	1.06
24	0.373	1.30×10^6	2.98×10^7	0.361	1.18×10^6	2.72×10^7	1.10
46	0.106	2.84×10^6	2.67×10^8	0.148	1.74×10^6	8.17×10^7	3.26

The cooling rate for classical SA was chosen to be $k = 8.9 \times 10^{-7}$ for LJ_{24} and $k = 1.30 \times 10^{-6}$ for LJ_{46}, based on the trend line in Fig. 3.12. The initial and final temperatures were chosen to be $T_i = 0.38$ and $T_f = 0.12$ for LJ_{24}, and $T_i = 0.40$ and $T_f = 0.01$ for LJ_{46}, positioning them to either side of the temperatures at which the heat capacity is (or might be, i.e. for LJ_{46}) observed to have a maximum.[42, 43]

For ACSA, the estimated SA parameters were combined with the previously developed heuristics to obtain estimated ACSA parameters. The ACSA parameters are summarized in Table 3.7, and the performance of both SA and ACSA algorithms is summarized in Table 3.8. This test case is more representative of a practical optimization, where the optimization parameters are not fully optimal, but are only roughly refined through experience with previous optimizations. Even with the heuristically predicted parameters ACSA outperforms classical simulated annealing with the efficiency of 3.26 (Table 3.8).

3.5 Conclusion

We have introduced a new global optimization method, which we call adaptive-cooling simulated annealing (ACSA), in which the SA cooling rate varies as the optimization proceeds, depending on the current heat capacity of the system. By using comparable cooling rates to traditional SA only when the system is annealing through the bottleneck region of phase space, and faster cooling rates at other times, the optimization proceeds more efficiently — more than compensating for the extra computational cost of evaluating the heat capacity.

The method has been demonstrated for small LJ clusters. It is comparable in efficiency to traditional SA for the smallest clusters, becoming more than twice as efficient for LJ_n clusters with $n > 20$, and with an efficiency that rises as the cluster size increases. It is reasonable to expect that the same trend will also apply to other systems as well: more complex energy landscapes will benefit more by automatically detecting the regions in which cooling can be done more rapidly.

The ACSA algorithm requires the choice of several additional parameters over classical SA. For

the specific case of LJ clusters, a set of heuristics were determined for obtaining reasonable optimization parameters, which have proven to result in improvements over SA. In the general case, ACSA can exactly reproduce the SA algorithm by using $k_s = k_f = k$ and $N_{\text{eq}} = N_{\text{prod}} = 0$. Thus, the SA performance is available as a lower bound. Then C_V^* can be chosen based on values observed during a trial observation, and k_f can be increased in order to gain efficiency, investing as much or as little effort into optimizing the parameters as is justified for the particular application.

In this proof-of-concept demonstration of the method, the algorithm was kept rather simple. It is easy to imagine extensions which will make the method even more efficient, however.

First of all, the heat capacities can be calculated on-the-fly during the cooling phase without introducing the prior equilibration and production phases. This will reduce the computational cost by approximately a factor of three. Another possible extension is an adaptive cooling schedule that uses a more general $K(T)$ function, rather than a simple switch between two discrete values. Lastly, it will likely be beneficial to use the density of states, rather than the heat capacity, to determine the cooling rate. The heat capacity provides a useful proxy for the number of states accessible at a particular temperature, but it would be more valuable to detect the number of *barriers* accessible. This is more closely related to the density of states. This property is more difficult to calculate, but could prove a more efficient statistical mechanical indicator for the optimal cooling rate. These extensions to the ACSA method will be explored in future investigations.

Chapter 4

An Entropy-Maximization Approach to the Automated Training Set Generation for Interatomic Potentials

The text of this chapter, will be submitted to J. Chem. Phys.: M. Karabin and D. Perez, "An Entropy-Maximization Approach to the Automated Training Set Generation for Interatomic Potentials"[1].

4.1 Introduction

The practical usefulness of atomistic simulations ultimately relies on the availability of interatomic potentials that are able to provide reliable energies and forces at a sufficiently affordable computational cost. Since electronic structure calculations using techniques such as density functional theory (DFT) are often prohibitively expensive, simplified empirical forms have been the norm, especially for molecular dynamical applications. Early empirical potentials were traditionally highly computationally efficient but often lacked in accuracy and transferability.

Over the last few years, the need to bridge the gap between empirical methods and direct electronic structure calculations has driven the explosive development of machine learning (ML) based approaches that aim to combine the accuracy of the electronic structure methods and the efficiency of the early simplified potentials. The two main components of the ML-based potentials are the representation of the atomic structures

with a set of generic descriptors that characterize local atomic environments and the use of large amounts of data to train complex non-linear functions of the descriptors to reproduce reference electronic structure calculations (energies, forces, stresses, etc.).

While ML-based potentials have proved able to capture subtle features of the training data, their ability to extrapolate to situations that markedly differ from those encountered during training remains limited[14]. Therefore, the accuracy of ML-based potentials is highly dependent on the choice of the training set, which should i) cover as much of the relevant configuration space as possible, and ii) remain sufficiently compact so that the cost of computing the reference values with quantum calculations and training the model remains affordable. Traditionally, training set generation has been a highly labor-intensive activity that relies on physical intuition in order to select the configurations that should be included.

Different approaches have been proposed to address the first objective using sampling strategies [44, 45, 46, 47, 48, 49], including evolutionary structural searches [50], normal mode sampling [51], and exploration of the potential energy surface using on-the-fly approximations of the target potential [44]. Training set configurations are also selected from DFT-MD simulations[52].

The second objective is often achieved by sub-sampling larger data sets. Possible approaches include random selection [53], binning-based sub-sampling to achieve uniform representation of relevant quantities like atomic forces [54], clustering in descriptors space to identify distinct groups [54]. Finally, a number of recent approaches incrementally include data to the training set based on whether the prediction of the properties of new configurations require extrapolation [55, 56, 57, 46, 58]. These approaches differ by the algorithm type and query strategy[57, 46].

In this manuscript, we unify the diversity and non-redundancy objectives in a simple local approach where the diversity of atomic environment within individual configurations (as measured by an entropy metric) is maximized subject to the constraint that the configurations do not contain unphysical configurations (i.e., overlapping atoms). This objective is embodied in a generic effective potential energy function whose low-lying local minima are good candidates for inclusion in a training set. Such minima are sampled using a simple annealing scheme that can be easily automated. Importantly, this effective energy is not meant as an approximation to the energy of the actual target system; instead it is an abstract construct that enables the creation of material-agnostic training sets. In this sense, our approach aims at creating a "universal" set of configurations that captures a very wide range of local environments and does not focus solely on low-lying energy structures. The large volume of configuration space covered entails a tradeoff between the size of the training set and the target accuracy, but the high transferability it affords is important to capture high-energy,

far from equilibrium effects that can occur in extreme conditions, such as under irradiation or at high pressures. A global approach where diversity maximization is carried out globally over the whole training set is currently in development and will be reported in an upcoming manuscript.

4.2 Methods

4.2.1 Entropy maximization approach

Implementing these ideas in practice requires first defining a set of atomic descriptors $\{\mathbf{q}\}$ that characterize the local environment of each atom, and then defining a measure of the diversity of the distribution of these descriptors within a configuration of atoms. A wide array of atomic descriptors have been proposed in the literature[59], as these form the inputs of many machine learning approaches that learn atomic energies. The method we propose is agnostic to the specific choice of descriptors so as long as they are differentiable functions of atomic positions. In the following, the set of m descriptors $q_{i,k}$ of the local atomic environment of atom i is arranged into a vector $\mathbf{q}_i$ of length m. As a measure of the diversity, we use the entropy of the m-dimensional distribution of atomic descriptors $S(\{\mathbf{q}\})$ contained in a given configuration of atoms, which is a natural choice in this case: it is maximized for a uniform distribution of descriptors and minimized for configurations where all environments are identical, i.e., it promotes diversity and penalizes redundancy. The effective energy we propose is therefore of the form:

$$V = E_{\text{repulsive}} - KS(\{\mathbf{q}\}) \tag{4.1}$$

where $E_{\text{repulsive}}$ is a short range repulsive term that penalizes very short distances between atoms (so as to enforce an excluded volume around each atom) and K is a so-called entropy scaling coefficient that tunes the relative importance of the entropy and of the repulsive contribution. Local minima of this function can therefore be expected to contain a high diversity of different environments without any two atoms being unphysically close. We postulate that low-lying minima of this effective potentials are therefore good targets for inclusion in a training set.

A number of approaches have been proposed to numerically estimate the entropy of a distribution of descriptors. In the following, we adopt a simple nonparametric form where the local density is approximated using the first neighbor distance (in descriptor space)[60]. In this case, the estimator is of the form:

$$S(\{q\}) = \frac{1}{n}\sum_{j=1}^{n} \ln\left(n \cdot \Delta q_j^{\min}\right) \tag{4.2}$$

where $\Delta q_j^{\min}$ is the nearest-neighbor distance in descriptor space, i.e., it is the minimal distance between the descriptor of atom j and those of any other atom in the configuration, i.e.,

$$\Delta q_j^{\min} \quad = \quad \min_l \sqrt{\Delta \mathbf{q}_{jl} \cdot \Delta \mathbf{q}_{jl}}. \tag{4.3}$$

and n is the number of atoms in the cell. This specific choice is not expected to be critical and other estimators could be used instead. Note, in this case we are calculating a descriptor entropy (an information theoretic entropy) rather than a thermodynamic entropy.

4.2.2 Computational details

The training set is incrementally constructed by adding independent local minima of the effective energy Eq. 4.1. As the effective potential (much like actual potentials) is very rough, a simple annealing procedure was introduced, as illustrated in Fig.4.1. Note that the aim is not to locate the global minimum of the effective energy but simply to avoid trapping in low entropy configurations. Our annealing procedure proceeds through a simultaneous ramping down of the temperature and ramping up of K. The goal is to initially favor a thorough shuffling of the atomic positions and avoid correlations between successive configurations by using a high temperature ($10,000$K) and no entropy bias. Entropy maximization is then gradually favored by linearly decreasing the temperature down to 0 and ramping up K to 1000 eV. The resulting configuration is then harvested and added to the training set. The cycle then simply repeats.

The maximal value of entropy scaling factor K was tuned so as not to overwhelm $E_{\text{repulsive}}$ while still providing a strong driving force for the maximization of the entropy. As shown in Fig. 4.2, increasing too far K yields configurations where some pairs of atoms become separated by very short distances. Large values of K also yield stiff effective potentials that are prone to instabilities during the MD annealing. We therefore settled on a maximal value of $K = 1000$ eV. Note that this choice depends on the number of atoms in the simulation cell, as the entropy so-defined is intensive, but the repulsive contribution is extensive.

The spatial scale of the problem was chosen to be representative of tungsten atoms, but the training

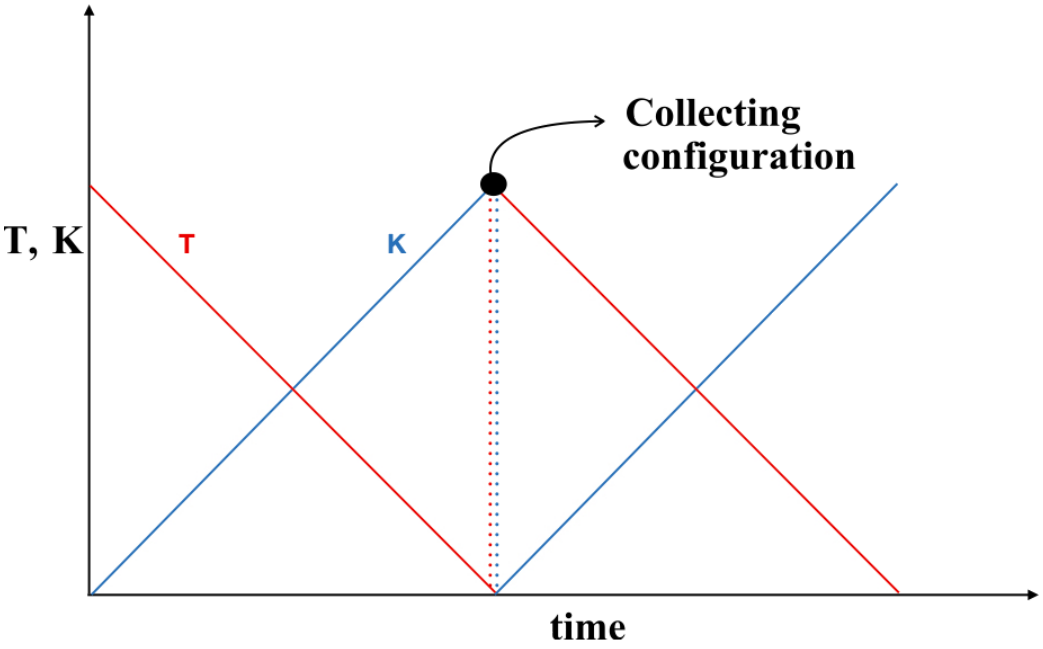

Figure 4.1: Schematic representation of the cyclic annealing procedure. See text for details.

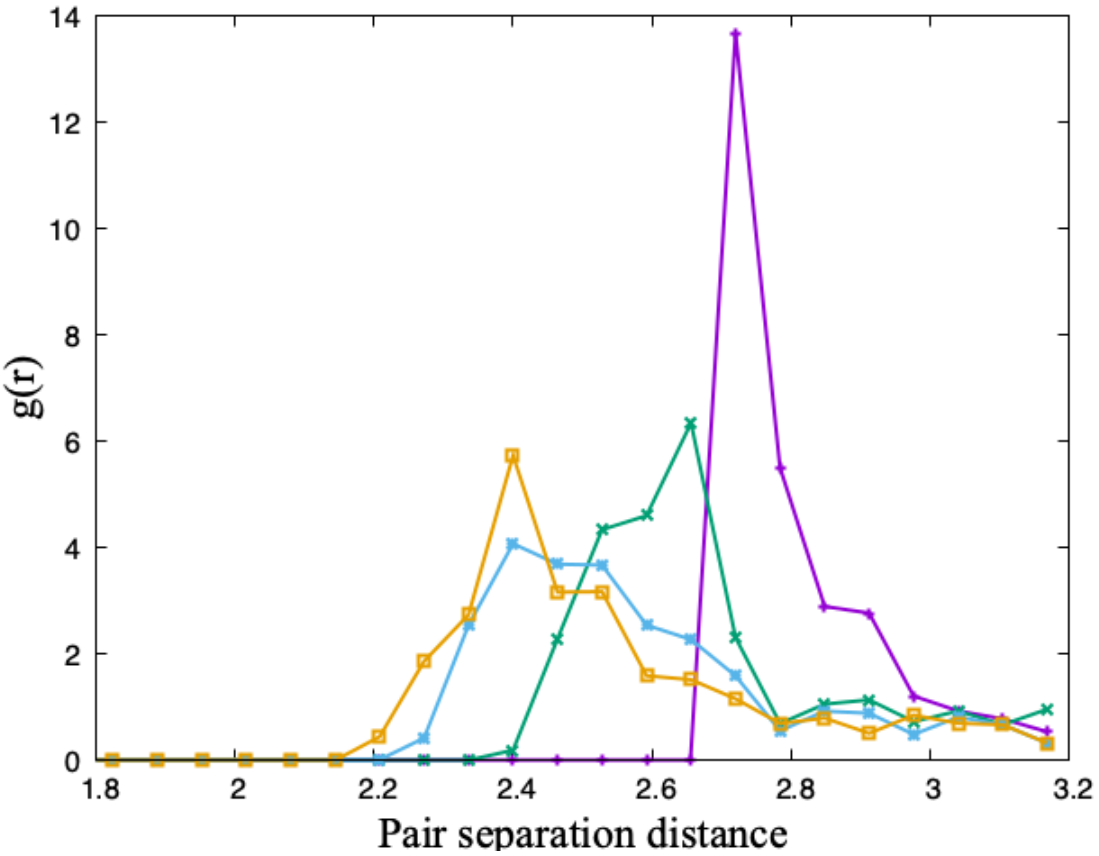

Figure 4.2: Radial distribution function for an annealed configuration obtained with $K = 0$ eV (purple), $K = 1000$ eV (green), $K = 2000$ eV (blue), and $K = 3000$ eV (yellow).

set is fully generic, and can therefore be rescaled as needed to describe other elements. In the following, we used cells containing $n = 39$ atoms with a volume of 9.54x9.54x14.31 Å. This corresponds to a density of 0.030 atoms/Å^3, as compared to a bulk BCC density of 0.062 atoms/Å^3 for tungsten. The number of atoms was chosen so that cubic-scaling DFT calculations would be affordable whereas the volume was chosen so that both high and low density regions could coexist within the same simulation cell, thereby creating configurations that contain to bulk, surfaces, and voids. While even larger volumes allowed for larger descriptor entropies because of additional opportunities to create complex atomic arrangements, local minima of the effective energy at low density tend to contain high proportion of 1D filament-like structures and of gas-like configurations. If such configurations are deemed relevant, a training set can constructed by combining a range of different cell sizes. This possibility will be explored a future study.

In the following, atomic environments were described in terms of the so-called bispectrum components originally developed in the context of the Gaussian Approximation Potentials (GAP) potentials[23], and then adopted by the SNAP approach [22]. These descriptors are invariants of an expansion of the the density of neighboring atoms around a central atom in terms of hyperspherical harmonics. They are attractive because they are rotationally and permutationally invariant, which facilitates the development of energy expressions that inherit from these same properties. Progressively higher-order components then capture increasingly fine details of the distributions of neighboring atoms. Details of the computation of the bispectrum components can be found in the original publications[23]. The results presented below used the first 6 bispectrum components to characterize each atomic environment.

In the following, $E_{\text{repulsive}}$ follows the form proposed by Clarke and Smith[61]:

$$E_{\text{repulsive}} = \sum_i \sum_j \frac{E_0}{n-m} \left[m \left(\frac{r_0}{r_{ij}} \right)^n - n \left(\frac{r_0}{r_{ij}} \right)^m \right] \tag{4.4}$$

with $E_0 = 1$ eV, $n = 8$, $r_0 = 2.7$ Å, and $m = 4$. The potential was truncated at $r = 2.71$Å, and shifted to zero at the cutoff so as to capture only the repulsive part of the potential. The results are not expected to be sensitive to the specific form of the repulsive potential, as its only purpose is to enforce excluded volumes around each atom.

4.3 Results

4.3.1 Characterization of the descriptor diversity

10,000 configurations of $n = 39$ atoms were generated using the procedure described above. A few representative configurations are shown in Fig. 4.3. The ensemble of these configurations is referred to as the "biased" dataset. As a point of comparison, we compare the results with a so-called "unbiased" reference dataset, where configurations were sampled from an MD simulation at a temperature of $T = 10,000$K with $K = 0$, i.e., without attempting to maximize the entropy but while enforcing excluded volume constraints. As expected, the average descriptor entropy of configurations in the biased set ($S \sim 4.4$) is larger than that of the unbiased set ($S \sim 3.2$).

The consequences of this increase in entropy can be appreciated by contrasting the distribution of individual descriptors over the biased and unbiased sets, as shown in Fig. 4.4 for the first bispectrum component. The distribution over the biased set is clearly much broader than its unbiased counterpart, which was the intended behavior. This shows that, even if the entropy maximization was applied locally to each configuration, the procedure yields broad distributions over the whole training set. Perhaps surprisingly, computing high ($>$ 6th) order descriptors shows that their distribution is also broadened in the biased set. A multiple correlation analysis demonstrates that this results from the fact that the descriptors are not mutually linearly independent; on average, we observe a correlation coefficient of about 0.7 for high-order descriptors against the first 6. Multiple correlation analysis is a measure of the linear relationship between one variable and the linear combination of the other variables in a set [62].

4.3.2 Error estimations on trained potentials: biased vs unbiased datasets

In order to quantify the quality of the generated training set, we consider an ML scenario where energies and forces are computed through a gaussian process regression (GPR) parameterized on a given training set [63, 64] as was done in the GAP approach [65, 14]. We consider the GPR to act as an interpolator that exactly reproduces reference data at training points. In this setting, the distance to the nearest training point (in descriptor space) is a simple surrogate for the error in predictions at test points. The shift and scale transformation that renders the distribution of each descriptors in the unbiased dataset mean free and unit variance was applied to both training and testing sets in order to uniformize the scales of each descriptors. In the following, the training sets contains 6000 randomly selected atomic environments, and testing sets 3000.

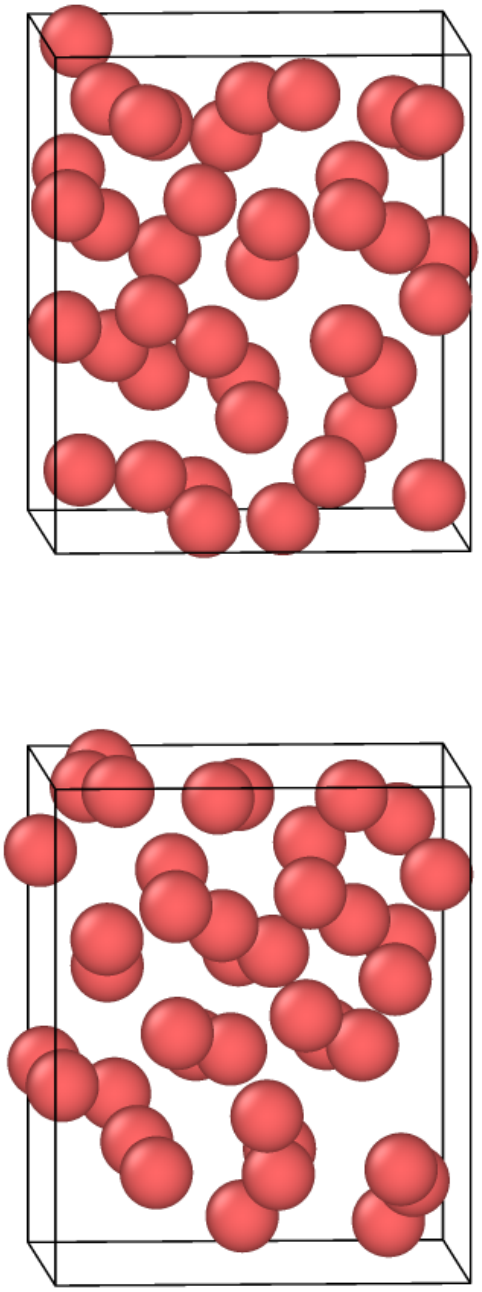

Figure 4.3: Configurations generated with the entropy maximization approach

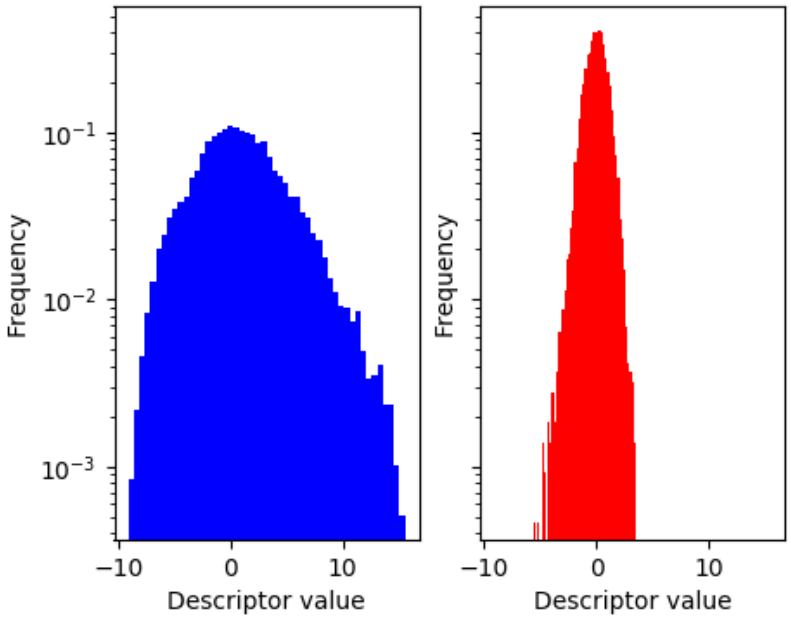

Figure 4.4: Distribution of the first descriptor. Left: biased dataset; Right: unbiased dataset.

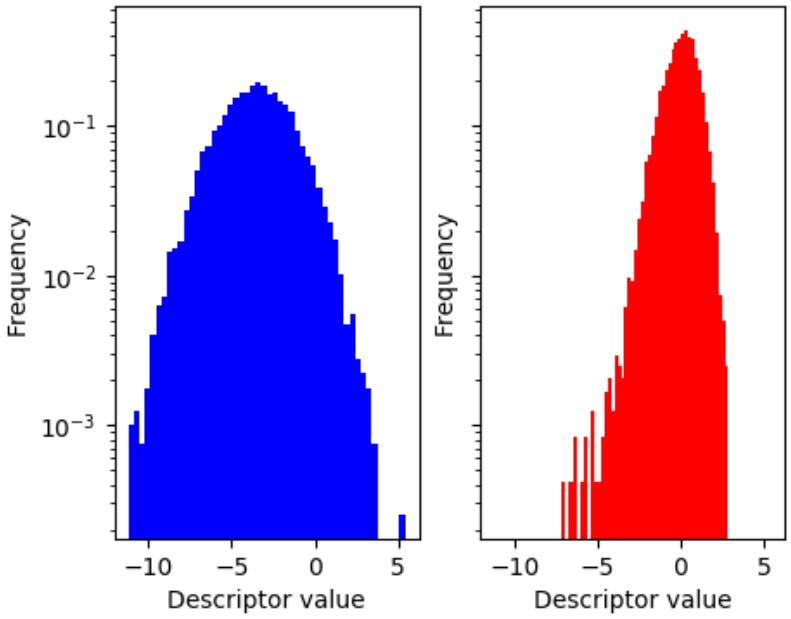

Figure 4.5: Distribution of the eight descriptor. Left: biased dataset; Right: unbiased dataset.

Results were averaged over 1000 random decompositions between testing and training sets. Distances were measured in the 6-dimensional space spanned by the biased descriptors.

This metric is used to first compare the quality of the unbiased and biased datasets. Table 4.1 shows the mean distances obtained using different combinations of training and testing sets. Training on the unbiased set performs well (i.e., the mean error is low) when testing points are also sampled from the unbiased set. The distances however become very large when testing points are sampled from the biased set. This is a reflection of the fact that the distribution of descriptors in the unbiased set has a relatively narrow support: the training distribution is therefore dense over the support (yielding small distances when the test points fall within the support) but very large distances when the test points fall outside of the support (i.e., when test points are sampled from the biased distribution). In this latter case, the GPR is extrapolating, leading to large distances. This illustrates an important tradeoff: a potential trained on a narrow set of configurations can be expected to do well when used on configurations close to this narrow set; it will however do poorly when departing from it. In contrast, the distances obtained from training on the biased set show little dependence on the nature of the testing set, as the GPR is not forced to extrapolate outside of the support of the training set. Another tradeoff apparent here is that the mean distance when training on the biased set and testing on the unbiased set are higher than those observed when training and testing on the unbiased set. This follows from the inverse relationship between the size of the support and the density of points in descriptor space at fixed number of training points. If one is aiming at transferability, the biased training set is however clearly favorable to the unbiased one.

4.3.3 Error estimations on trained potentials: biased vs hand-crafted datasets

A more stringent test of our approach is to compare the biased training set with an "hand-crafted" dataset that was used to train potentials reported in the literature. To this end, we select a well established training set that was used in the development of a number of recent potentials for tungsten[14].

Table 4.2 reports the results of the distances to the nearest training point for training and testing sets drawn from the hand-crafted and biased sets. The results are largely similar to that observed for the testing set. Training and testing from the hand-crafted set yields low errors as the support of both training and testing distributions is very narrow (c.f. purple histogram in Fig. 4.6), but these increase dramatically upon switching the test set to the unbiased set, again because extrapolation is then required (c.f. the very long tail in the green distribution in Fig. 4.6). In contrast, training from the biased set yields results that are similar for both testing sets (c.f. blue and red histograms in Fig. 4.6). These results suggest that the automated training set should be

Table 4.1: Error estimations on trained potentials: biased vs unbiased datasets

Training set	Testing set	Mean	Median	Max
Unbiased	Unbiased	0.2294	0.1827	5.9339
Unbiased	Biased	4.6949	3.5311	32.0983
Biased	Unbiased	0.9053	0.9068	2.3402
Biased	Biased	0.8541	0.7846	5.7246

Table 4.2: Error estimations on trained potentials: biased vs hand-crafted datasets

Training set	Testing set	Mean	Median	Max
Hand-crafted	Hand-crafted	0.2263	0.1562	7.6514
Hand-crafted	Biased	3.6680	2.3153	36.4918
Biased	Hand-crafted	0.9923	1.0014	7.9049
Biased	Biased	0.8541	0.7846	5.7246

competitive with the hand-crafted set as it contains a more diverse distribution of atomic environments.

Close analysis reveals that this characterization comes with caveats. Indeed, as shown in Fig. 4.7, the distribution of descriptors is in general significantly wider in the biased set than in the hand-crafted set. However, the distribution of some descriptors in the hand-crafted set is strongly peaked, as it contains a high proportion of crystalline local environments. In some cases, the peak falls into a region where the density descriptors in the biased dataset is low, c.f. Fig. 4.8, which can limit the accuracy of predictions carried out using the biased set alone for training. This translates into an increasing errors when testing with the hand-crafted states as the space of descriptors in which the GPR interpolation is carried out increases to tens or hundreds of dimensions. Note however that even in this case, the errors remain below that of training with the hand-crafted set and testing with the biased set. This limitation could potentially be addressed by increasing the size of the biased space or by explicitly favoring high-symmetry local order.

This observation illustrates the tradeoffs discussed above: if one seeks an highly accurate potentials that is valid in a small region of the possible configuration space of the problem (e.g. in BCC crystalline configurations), a narrow but tailored training set is likely to perform better; on the other hand, if transferability is paramount, an approach that explicitly favors diversity as the one proposed here is highly beneficial. In practice, these two extremes can be bridged to achieve both high accuracy in low-energy states and transferability to higher-energy configurations.

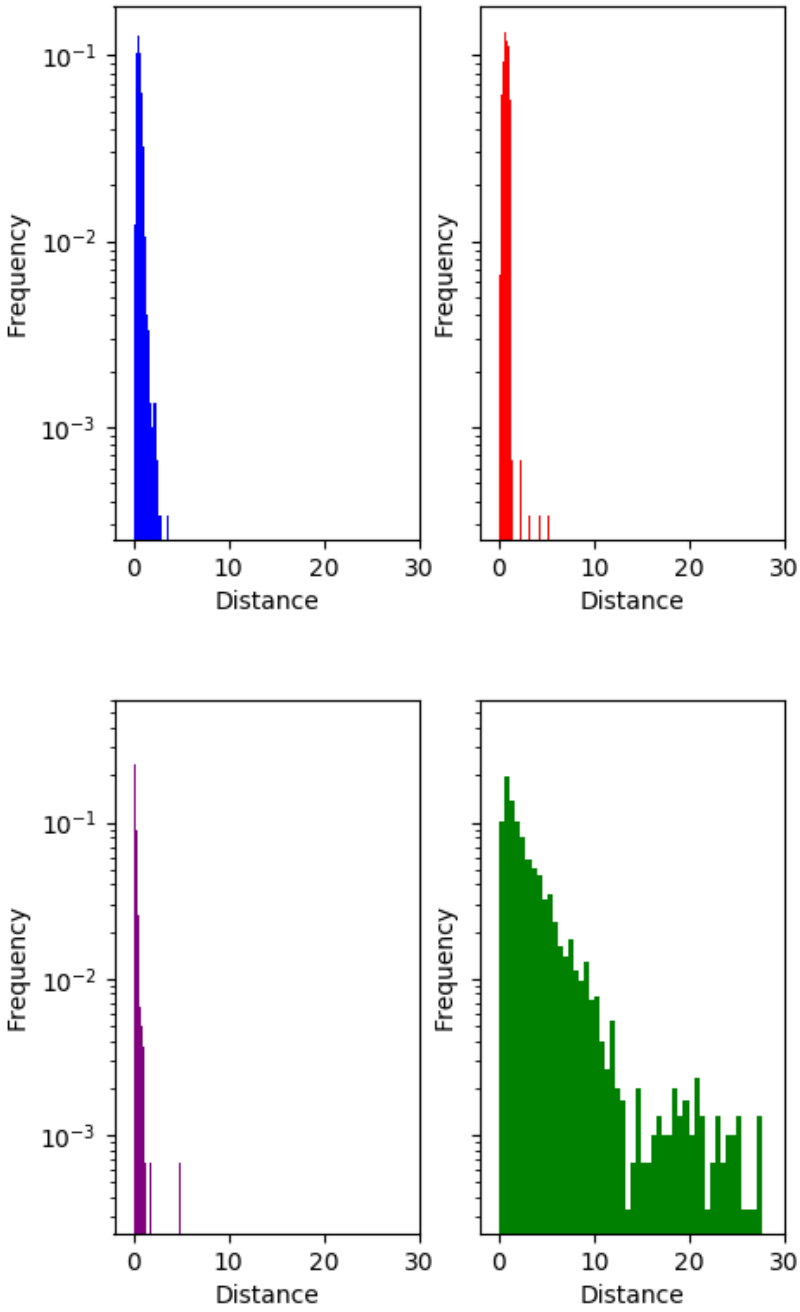

Figure 4.6: Distribution of the distances to the closest training point for different combinations of training and testing sets. Training from the biased set and testing from the hand-crafted set (red); training and testing from the biased sets (blue); training and testing from the hand-crafted set (purple); training from the hand-crafted set and testing from the biased set (green).

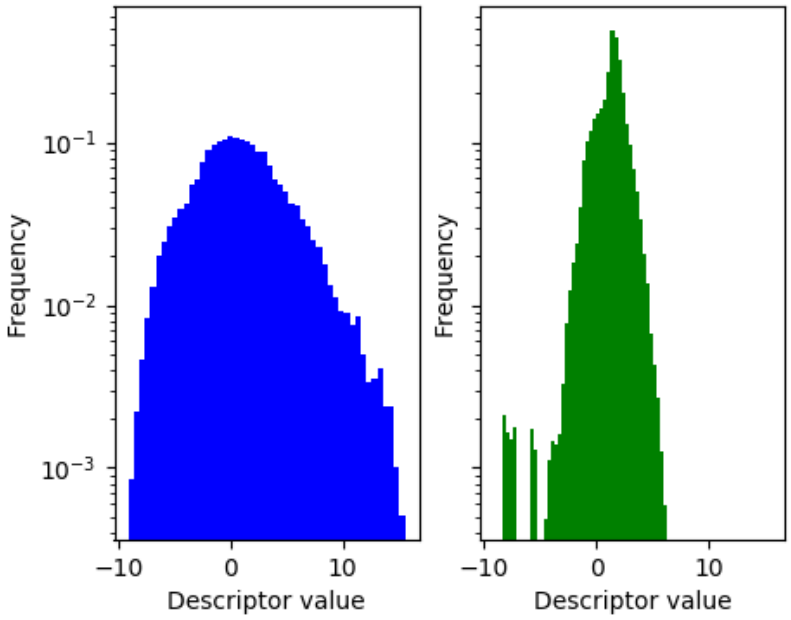

Figure 4.7: Distribution of the first descriptor. Left: biased dataset; Right: hand-crafted dataset.

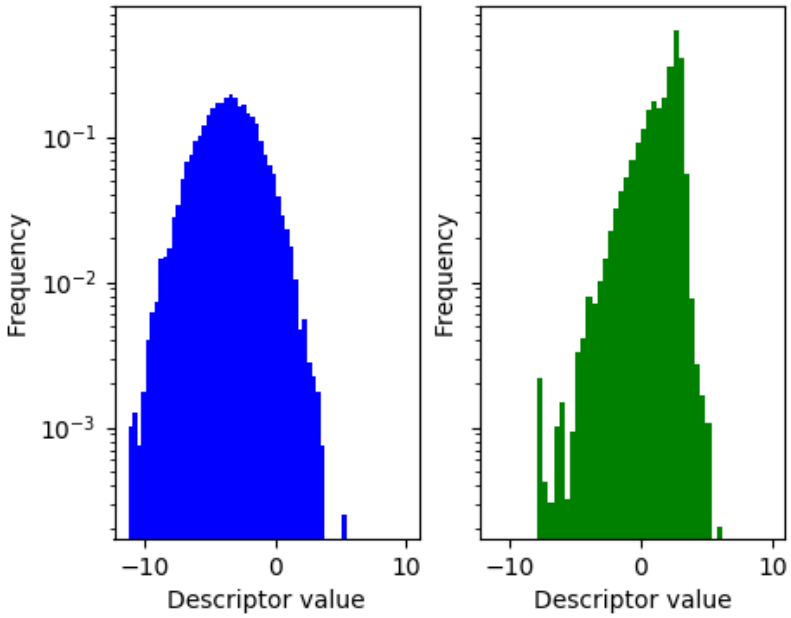

Figure 4.8: Distribution of the eight descriptor. Left: biased dataset; Right: hand-crafted dataset.

4.4 Conclusions

We introduced a sampling-based approach for the automated training set generation of interatomic potentials. Configurations are generated by sampling low-lying minima of an effective potential energy function that explicitly favors the diversity of the local atomic environment through an entropy maximization process. The generated training set is shown to be more diverse than that generated by a random sampling procedure and even compared to hand-crafted sets used in state-of-the-art machine-learned potentials, which promises improved transferability. Extensions to global entropy-maximization over the whole training set (in contrast to the local configuration-by-configuration optimization presented here) is in development and will be reported in an upcoming publication.

4.5 Acknowledgements

We thank Mitchell Wood and Nicholas Lubbers for stimulating discussions. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. Los Alamos National Laboratory is operated by Triad National Security LLC, for the National Nuclear Security administration of the U.S. DOE under Contract No. 89233218CNA0000001.

Chapter 5

Conclusions and Discussion

Statistical mechanics tools have proven to be a highly powerful way to improve computational chemistry tools and circumvent its limitations. In a variety of previously, well known methods, these tools have improved the simulation of long-time trajectories, sampling of rare events, simulation of the larger system sizes, and much more. They have proven to be highly useful and deserve a further exploration. This dissertation presented two new methods: adaptive-cooling simulated annealing and an entropy maximization-based approach for the automated training set generation, in which statistical mechanics was used as a main tool to improve the original approaches. These original approaches are classical simulated annealing and a manual generation of the training sets for potential fitting.

Adaptive-cooling simulated annealing uses instantaneous values of the heat capacities as a signal to make a decision on a cooling rate based on our adaptive dual-cooling rate approach. The efficiency of this method over the classical simulated annealing calculated for the Lennard-Jones clusters is increasing as the cluster size increases. For the cluster with 36 atoms the efficiency is 2.67 and is expected to increase for larger clusters. This methods depends on the 7 initial parameters that need to be set before the optimization starts, and it was shown that these parameters can be estimated based on the prior knowledge from the classical simulated annealing optimizations of the Lennard-Jones clusters. A few improvements have been suggested as possible extensions[13].

Another method presented is an entropy maximization-based approach for automated training set generation for machine learning-based potentials that makes use of the entropy bias to generate a training set with a diverse set of configurations. It has been shown that this approach improves the transferability and is not as labor-intensive when compared to the manually generated hand-crafted data set from the literature

resources used for developing potentials for tungsten. A more global approach is under development in which a modification to the current entropy maximization-based method is proposed to maximize the diversity of the configurations across the whole training set rather than more locally for the atomic environment.

Current computational tools still have much room for improvement and many more limitations to overcome, and statistical mechanics has proven to be great tool that can be used to circumvent these limitations and design even more powerful computational methods.

Appendices

Appendix A Characterization of the SNAP descriptors

The representation of atomic neighborhood is affected by the choice and the quality of the descriptors, which should be invariant under rotation, translation, and permutation of atoms of the same chemical species, as was described in the methods section. To analyze the quality of the descriptors, which in this case are bispectrum components, the intrinsic dimension estimation and correlation analysis were performed. Intrinsic dimension estimation gives information about how many descriptors are needed to describe the system, and correlation analysis gives information about the correlation between descriptors, which ideally should be uncorrelated or have as minimal correlation as possible.

A.1 Intrinsic dimension estimation

The intrinsic dimension (ID) estimation was performed to find the minimum number of variables (descriptors) that could be used to describe the system. It is a common dimensionality reduction technique used to understand a highly dimensional data.

The intrinsic dimension, d, as was defined in the methods section as:

$$-\frac{log(1-F(m))}{log(m)} = d \tag{1}$$

where, for each descriptor, the two closest sets of descriptors are identified, and m is calculated as the ratio of the two distances to these descriptors. The function F(m) is the cumulative probability distribution function of these ratios, as described in the methods section. Consequently, $\log(1-F(m))$ was plotted as a function of $\log(m)$. These results are plotted in Figure 1, for the case where K=1000. The data points $(\log(m_i), -\log(1-F(m_i)))|i = 1,...,N$ were fit using least-squares linear regression, where $y = d \cdot x$, and the slope of the line corresponds to the intrinsic dimension d.

Ideally, $\log(1-F(m))$ should scale linearly with $\log(m)$, which would mean that descriptors are uncorrelated or low-correlated and are indispensable for describing the system.

Original data set consists of 200 descriptors; where for each of the descriptors the distances to the two closest neighbors were calculated to get $\log(m)$ and $\log(1-F(m))$.

The results show that in case when K = 1000 (Fig. 1), the intrinsic dimension is 10. 7, which was calculated as a slope of the line fitted using linear regression. Intrinsic dimension of 10.7 means that 10 descriptors are needed for the simulation to describe the system. This is a fairly low dimension considering

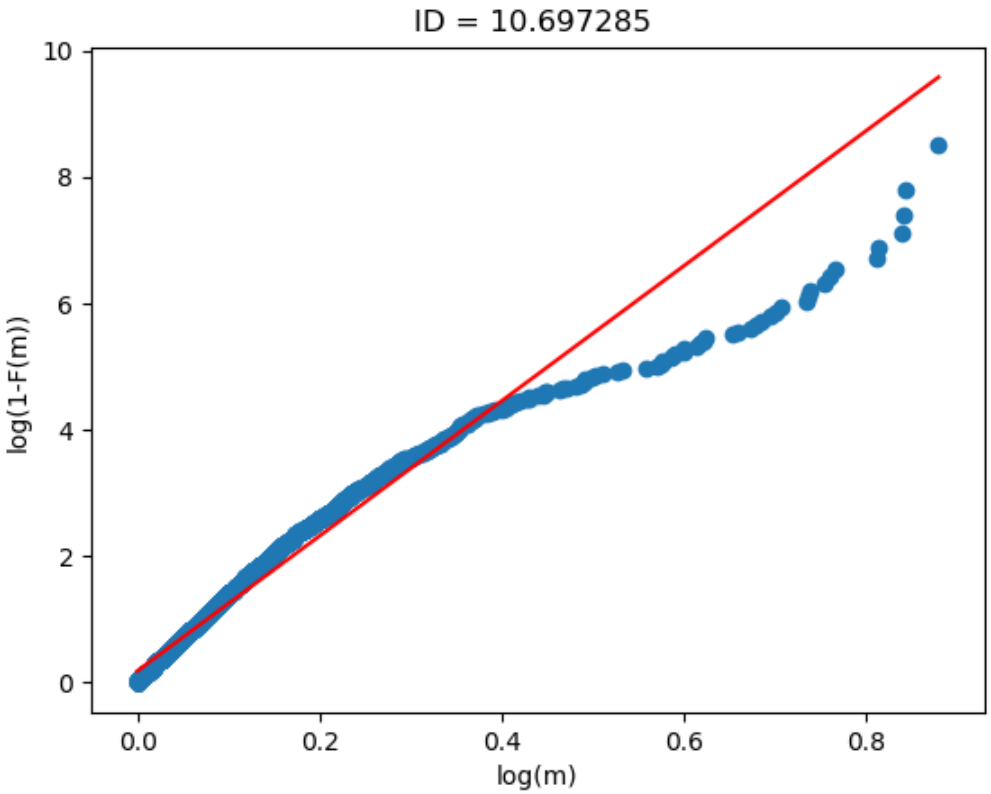

Figure 1: Intrinsic dimension estimation with the entropy scaling coefficient K = 1000.

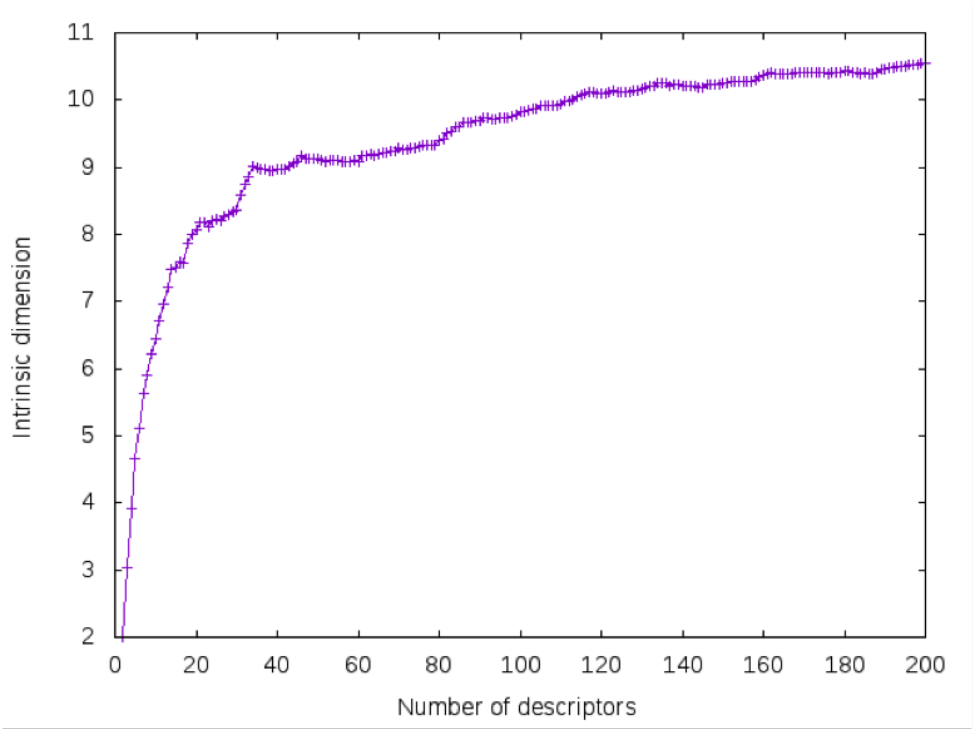

Figure 2: Dependence of intrinsic dimension on a number of descriptors, with the entropy scaling coefficient K = 1000.

that the total number of descriptors used was 200. Also, the results show that with higher order bispectrum components data becomes noisier and the slope decreases, which mean that higher order descriptors are more correlated than lower order descriptors.

Further analysis of the intrinsic dimension (Fig. 2) was performed to analyze the change in the intrinsic dimension with the number of descriptors. The original data set is the same as the one used in the previous test and consists of the same 200 descriptors. The analysis was performed by starting with a minimal set of descriptors (3 bispectrum components), and then calculating the intrinsic dimension every time the new descriptor was added to the given data set.

The results shows that when calculating intrinsic dimension for a set of descriptors, by adding one descriptor at a time to the given set, intrinsic dimension increases strongly only for the first 10-15 descriptors. This means that we are not getting that much more new information when using more than 10-15 descriptors. Ideal case would be to have a close to linear relationship between the intrinsic dimension and descriptors.

A fairly low intrinsic dimension can indicate two effects: first, the data is truly low dimensional in some basis, or two, the descriptors are highly correlated, so that the space they span increases only slowly.

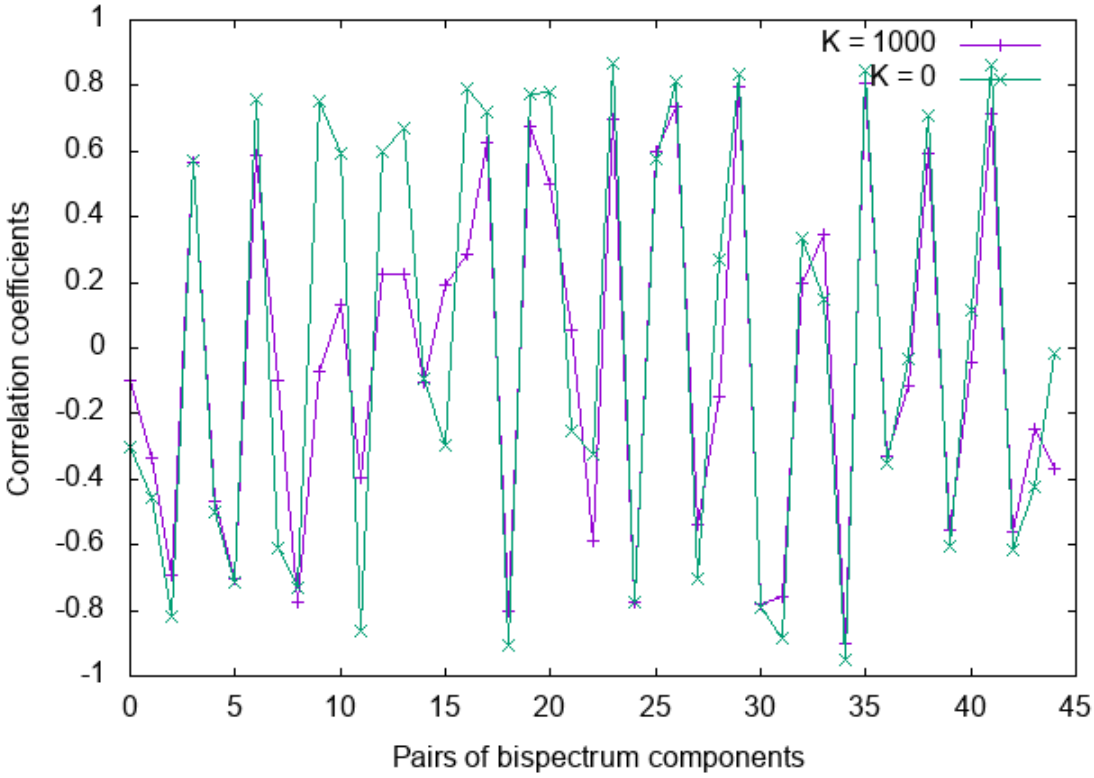

Figure 3: Pearson correlation coefficients for the pairs of bispectrum components B_{ij} with the entropy scaling coefficient K = 1000 and K = 0.

Therefore, the next test was Pearson correlation coefficients analysis.

A.2 Pearson correlation coefficients

Pearson correlation coefficients analysis (Fig. 3.) was performed to analyze the correlation between the bispectrum components. Each of the coefficients shows the strength of the linear relationship between a pair of the bispectrum components. The stronger the relationship is, the closer correlation coefficients are to 1 or -1, and 0 means that there is no linear relationship between that pair of variables.

For this analysis the first 10 descriptors of the same set of bispectrum components were used. The correlation coefficients were calculated for every pair of bispectrum components B_{ij} , where i $\neq$j, and $B_{ij} = B_{ji}$ (Fig. 3).

The results show that bispectrum components are highly correlated, as many of the data points are close to 1 and -1. Descriptors are even more correlated in the case when entropy scaling coefficient K = 0.

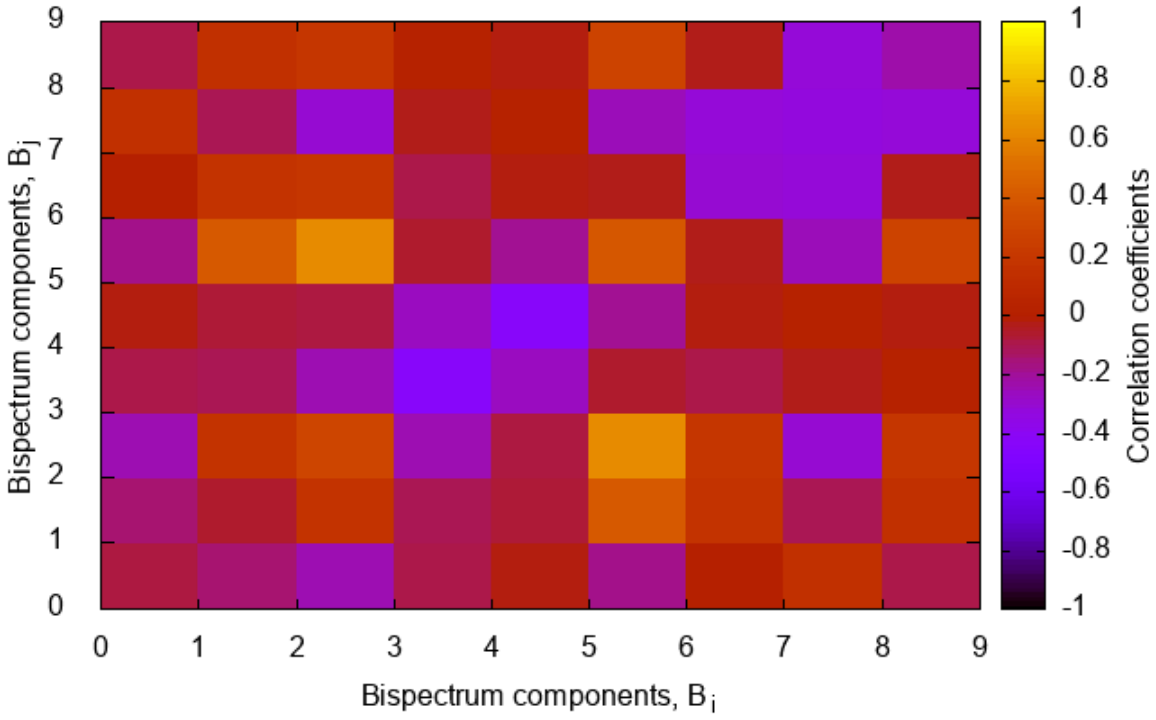

Figure 4: Pearson correlation coefficients for the pairs of bispectrum components B_i and B_j with the entropy scaling coefficient K = 1000.

The average absolute value of the correlation coefficient is 0.46, when K=1000, and 0.59 when K=0. Ideally, bispectrum components should be uncorrelated or have as low correlation as possible. High correlation between the descriptors means that these descriptors do not add any new information when describing a system.

The results from Fig. 3 were also plotted as a heat map separately for different entropy scaling coefficients (Fig. 4,Fig. 5) to show which Pearson correlation coefficients correspond to which pairs of the bispectrum components.

Results from the Fig. 4 show that there are three pairs of bispectrum components $(B_{4,2}, B_{4,9}, B_{5,6})$ which have the correlation coefficients higher than 0.8 when K = 1000. On the contrary, when K = 0 there are 10 pairs of the bispectrum components with the correlation coefficients higher than 0.8. These pairs of the bispectrum components with the high correlation coefficients correspond to the higher order descriptors, which is related to the fact that the calculation of the higher order descriptors requires the information from

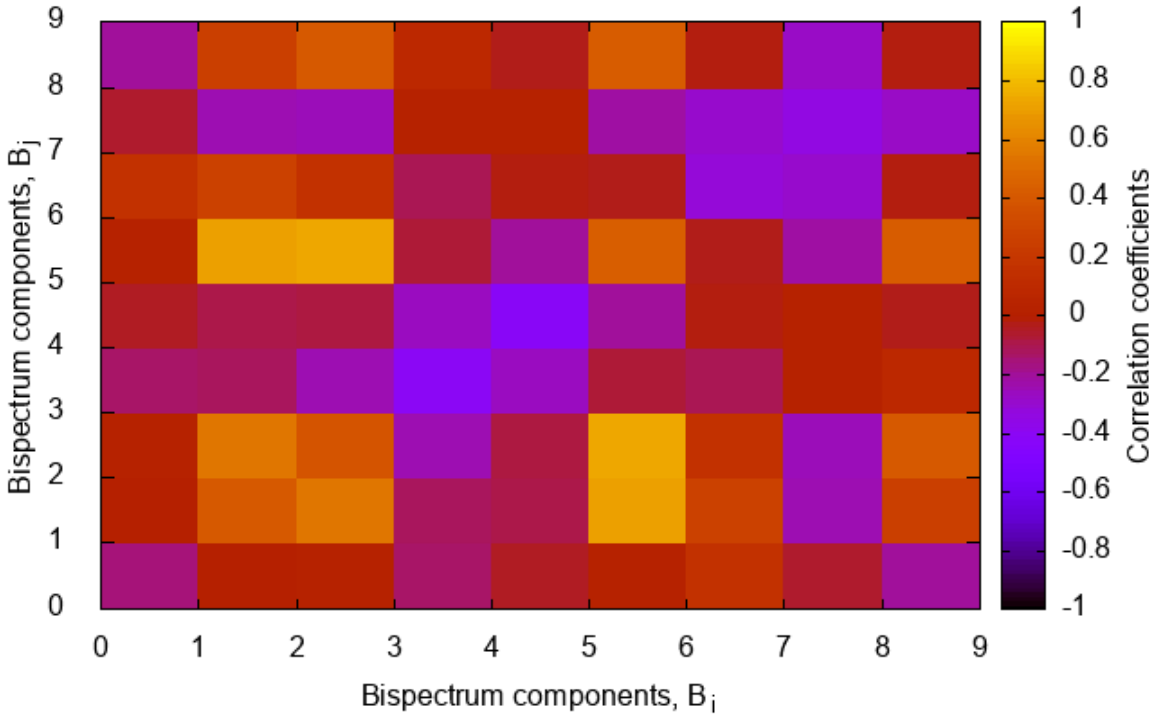

Figure 5: Pearson correlation coefficients for the pairs of bispectrum components B_i and B_j with the entropy scaling coefficient K = 0.

the lower order descriptors.

To get more information about the correlation between the bispectrum components, the multiple correlation analysis was performed.

A.3 Multiple correlation results

Multiple correlation analysis is a measure of the linear relationship between one variable and the linear combination of the other variables in a set [62]. Meaning how well one variable can be predicted using a linear function of a set of other variables. It is used to determine the relationship between a few variables at a time, whereas Pearson correlation analysis was used to determine the relationship between two variables at a time.

The multiple correlation coefficients are between 0 and 1, where 0 means there is no correlation between the variables, and 1 means a high correlation and so the variable can be perfectly explained with a linear combination of the other variables.

The multiple correlation analysis was performed by determining the correlation between every descriptor and a set of other 199 descriptors at a time. Descriptors were reordered to show the number of descriptors that give the lowest correlation.

The analysis (Fig. 6) shows that the bispectrum components become highly correlated at a very low number of descriptors, which is close to 10 descriptors (reordered, Fig. 6). This means that every other descriptor except these 10 descriptors bring much less information in describing the system, whereas the computational cost of calculating the descriptors keeps increasing. 10 low correlated descriptors out of the total of 200 descriptors is a fairly low number.

Ideally, the descriptors should be uncorrelated or have as low correlation as possible to describe the system well. This result is consistent with the results and conclusions from intrinsic dimension estimation and Pearson correlation analysis.

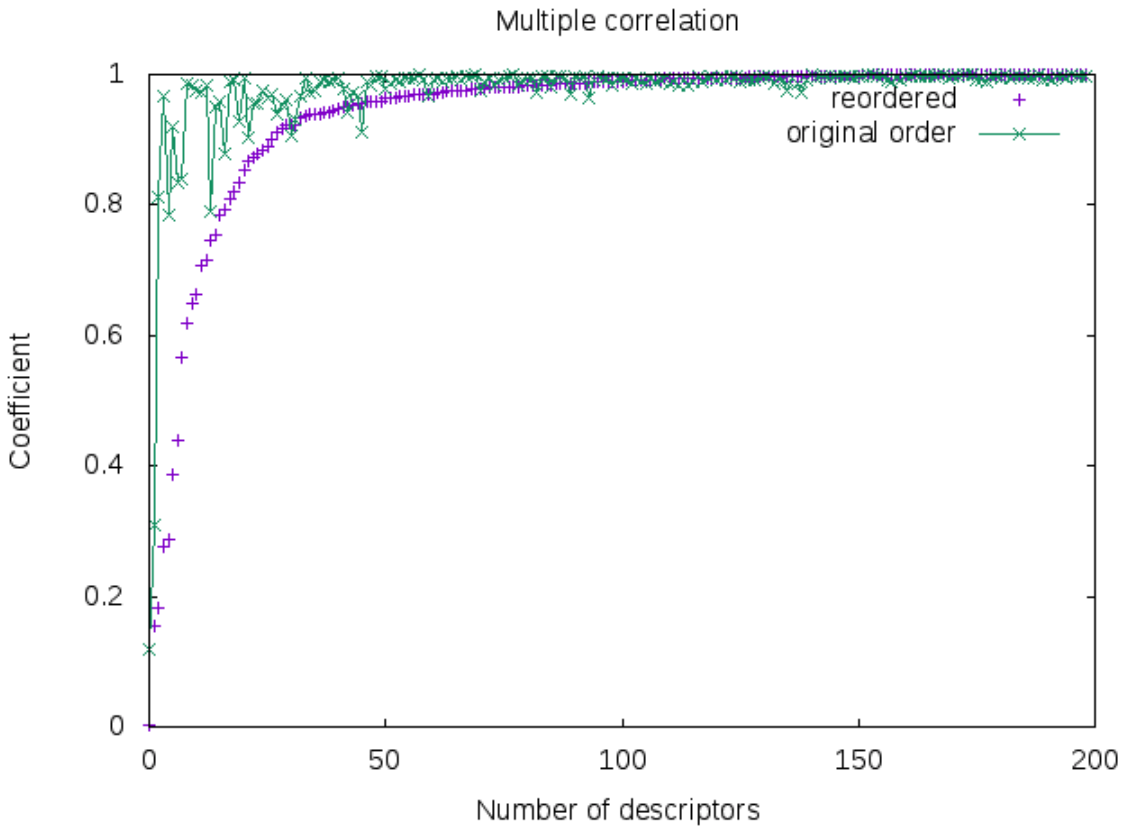

Figure 6: Dependence of the multiple correlation coefficients on the number of descriptors for the original and reordered set of descriptors.

Bibliography

[1] Mariia Karabin and Danny Perez. An entropy-maximization approach to automated training set generation for interatomic potentials. *arXiv*, 2020.

[2] Yakir Forman and Maria Cameron. Modeling aggregation processes of lennard-jones particles via stochastic networks. *Journal of Statistical Physics*, 168(2):408–433, May 2017.

[3] Sigurd Schelstraete and Henri Verschelde. Finding minimum-energy configurations of lennard-jones clusters using an effective potential. *The Journal of Physical Chemistry A*, 101(3):310–315, 1997.

[4] Danny Perez, Blas P. Uberuaga, Yunsic Shim, Jacques G. Amar, and Arthur F. Voter. Chapter 4 accelerated molecular dynamics methods: Introduction and recent developments. volume 5 of *Annual Reports in Computational Chemistry*, pages 79 – 98. Elsevier, 2009.

[5] L. Wu and Y. Wang. Introduction to simulated annealing algorithms for the computation of economic equilibrium. *Computational Economics*, 12, 1998.

[6] Ron O. Dror, Morten Ø. Jensen, David W. Borhani, and David E. Shaw. Exploring atomic resolution physiology on a femtosecond to millisecond timescale using molecular dynamics simulations. *J Gen Physiol.*, 135, 2010.

[7] Arthur F. Voter. Hyperdynamics: Accelerated molecular dynamics of infrequent events. *Phys. Rev. Lett.*, 78:3908–3911, May 1997.

[8] Mads R. Sorensen and Arthur F. Voter. Temperature-accelerated dynamics for simulation of infrequent events. *The Journal of Chemical Physics*, 112(21):9599–9606, 2000.

[9] Arthur F. Voter. Parallel replica method for dynamics of infrequent events. *Phys. Rev. B*, 57:R13985–R13988, Jun 1998.

[10] Tatiana Maximova, Ryan Moffatt, Buyong Ma, Ruth Nussinov, and Amarda Shehu. Principles and overview of sampling methods for modeling macromolecular structure and dynamics. *PLOS Computational Biology*, 12(4):1–70, 04 2016.

[11] A. Leach. *Molecular Modeling: principles and applications*. Harlow: Pearson Prentice Hall, 2001.

[12] A. Ghoufi, F. Goujon, V. Lachet, and P. Malfreyt. Multiple histogram reweighting method for the surface tension calculation. *The Journal of Chemical Physics*, 128(15):154718, 2008.

[13] Mariia Karabin and Steven J. Stuart. Simulated annealing with adaptive cooling rates. *arXiv*, 2020.

[14] M. A. Wood, M. A. Cusentino, B. D. Wirth, and A. P. Thompson. Data-driven material models for atomistic simulation. *Phys. Rev. B*, 99:184305, May 2019.

[15] B. J. Alder and T. E. Wainwright. Phase transition for a hard sphere system. *J. Chem. Phys.*, 27, 1957.

[16] B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. i. general method. *J. Chem. Phys.*, 31, 1959.

[17] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.*, 76, 1982.

[18] N. Siddique and H. Adeli. Simulated annealing, its variants and engineering applications. *International Journal on Artificial Intelligence Tools*, 25:06, 2016.

[19] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21, 1953.

[20] W. Press, S. Teukolsky, W. Vetterling, and P. Flannery. *Numerical Recipes in C++: The Art of Scientific computing*. Cambridge University Press, 2002.

[21] John A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7 (4):308–313, 1965.

[22] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, Mar 2015.

[23] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.*, 104:136403, Apr 2010.

[24] S. Ledesma, G. Avina, and R. Sanchez. *Simalated Annealing*, chapter 20, pages 401–420. InTech, 2008.

[25] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[26] D. Frenkel and B. Smit. *Understanding molecular simulation: from algorithms to applications*. Academic Press, 2012.

[27] R.Harada, T. Nakamura, and Y. Shigeta. A fast convergent simulated annealing algorithm for protein - folding: Simulated annealing outlier flooding (sa-oflood) method. *Bulletin of the Chemical Society of Japan*, 89:367, 2016.

[28] H. H. Szu and R. L. Hartley. Non-convex optimization by fast simulated annealing. *Proceedings of the IEEE*, 75:1538–1540, 1987.

[29] H. Lee and D. Arditi. An advanced stochastic time-cost tradeoff analysis based on a cpm-guided multi-objective genetic algorithm. *Computer-Aided Civil and Infrastructure Engineering*, 30:824–842, 2015.

[30] L. Vincenzi and M. Savoia. Coupling response surface and differential evolution for parameter identification problems. *Computer-Aided Civil and Infrastructure Engineering*, 30:376–393, 2015.

[31] F. Shabbir and P. Omenzetter. Particle swarm optimization with sequential niche technique for dynamic finite element model updating. *Computer-Aided Civil and Infrastructure Engineering*, 30:359–375, 2015.

[32] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76:60–68, 2001.

[33] L. Ingber. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12:967–973, 1989.

[34] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34:111–124, 1986.

[35] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and bayesian restoration of images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[36] P. Van Laarhoven, E. Aarts, and J. Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40:113–125, 1992.

[37] Y. Nourani and B. Andersen. A comparison of simulated annealing cooling strategies. *J. Phys. A: Math. Gen.*, 31:8373–8385, 1998.

[38] D. Fodorean, L. Idoumghar, A. N'diaye, D. Bouquain, and A. Miraoui. Simulated annealing algorithm for the optimisation of an electrical machine. *IET Electr. Power Appl.*, 6:735–742, 2012.

[39] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18:29–57, 1993.

[40] D. J. Wales, J. P. K. Doye, A. Dullweber, M. P. Hodges, F. Y. Naumkin, F. Calvo, J. Hernández-Rojas, and T. F. Middleton. The cambridge cluster database. `http://www-wales.ch.cam.ac.uk/CCD.html`, 1997.

[41] Gropp, Lusk, and Skjellum. *Using MPI*. MIT Press, 1994.

[42] Lívia B. Pártay, Albert P. Bartók, and Gábor Csányi. Efficient sampling of atomic configurational spaces. *The Journal of Physical Chemistry B*, 114:10502–10512, 2010.

[43] D. D. Frantz. Magic numbers for classical lennard-jones cluster heat capacities. *J. Chem. Phys.*, 102:3747, 1995.

[44] V. L. Deringer, C. J. Pickard, and G. Csanyi. Data-driven learning of total and local energies in elemental boron. *Phys. Rev. Lett.*, 120, 2018.

[45] V. L. Deringer, D. M. Proserpio, G. Csanyi, and C. J. Pickard. Data-driven learning and prediction of inorganic crystal structures. *Faraday Discuss.*, 211, 2018.

[46] Evgeny V. Podryabinkin, Evgeny V. Tikhonov, Alexander V. Shapeev, and Artem R. Oganov. Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning. *Phys. Rev. B*, 99:064114, Feb 2019.

[47] S. Hajinazar, J. Shao, and A. N. Kolmogorov. Stratified construction of neural network based interatomic models for multicomponent materials. *Phys. Rev. B*, 95, 2017.

[48] S. Chmiela, H. Sauceda, K. Müller, and A. Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nature Communications*, 9(3887), 2018.

[49] Kyuhyun Lee, Dongsun Yoo, Wonseok Jeong, and Seungwu Han. Simple-nn: An efficient package for training and executing neural-network interatomic potentials. *Computer Physics Communications*, 242:95 – 103, 2019.

[50] H. Chan, B. Narayanan, M. Cherukara, F. Sen, K. Sasikumar, S. Gray, M. Chan, and S. Sankaranarayanan. Machine learning classical interatomic potentials for molecular dynamics from first-principles training data. *The Journal of Physical Chemistry C*, 123, 2019.

[51] J. S. Smith, O. Isayev, and A. E. Roitberg. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chem. Sci.*, 8:3192–3203, 2017.

[52] Wonseok Jeong, Kyuhyun Lee, Dongsun Yoo, Dongheon Lee, and Seungwu Han. Toward reliable and transferable machine learning potentials: Uniform training by overcoming sampling bias. *The Journal of Physical Chemistry C*, 122, 2018.

[53] V. Botu, R. Batra, J. Chapman, and R. Ramprasad. Machine learning force fields: Construction, validation, and outlook. *The Journal of Physical Chemistry C*, 121(1):511–522, 2017.

[54] T. Huan, R. Batra, J. Chapman, S. Krishnan, L. Chen, and R. Ramprasad. A universal strategy for the creation of machine learning-based atomistic force fields. *npj Computational Materials*, 3(37), 2017.

[55] J Behler. Representing potential energy surfaces by high-dimensional neural network potentials. *Journal of Physics: Condensed Matter*, 26(18):183001, apr 2014.

[56] Søren L. Frederiksen, Karsten W. Jacobsen, Kevin S. Brown, and James P. Sethna. Bayesian ensemble approach to error estimation of interatomic potentials. *Phys. Rev. Lett.*, 93:165501, Oct 2004.

[57] Evgeny Podryabinkin and Alexander V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Computational Materials Science*, 140:171 – 180, 2017.

[58] Konstantin Gubaev, Evgeny V. Podryabinkin, Gus L.W. Hart, and Alexander V. Shapeev. Accelerating high-throughput searches for new alloys with active learning of interatomic potentials. *Computational Materials Science*, 156:148 – 156, 2019.

[59] Wiktor Pronobis, Alexandre Tkatchenko, and Klaus-Robert Müller. Many-body descriptors for predicting molecular properties with machine learning: Analysis of pairwise and three-body interactions in molecules. *Journal of Chemical Theory and Computation*, 14(6):2991–3003, 2018. PMID: 29750522.

[60] J. Beirlant, EJ Dudewicz, László Györfi, and István Dénes. Nonparametric entropy estimation. an overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997.

[61] Clarke and Smith. Short range effective potentials for ionic fluids. *J Chem Phys*, 84:2290, 1986.

[62] J. Cohen and P. Cohen. *Applied multiple regression/correlation analysis for the behavioral sciences*. Hillsdale (NJ): Erlbaum, 1983.

[63] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[64] C. Handley and J. Behler. Next generation interatomic potentials for condensed systems. *The European Physical Journal B*, 87, 2014.

[65] Wojciech J. Szlachta, Albert P. Bartók, and Gábor Csányi. Accuracy and transferability of gaussian approximation potential models for tungsten. *Phys. Rev. B*, 90:104108, Sep 2014.